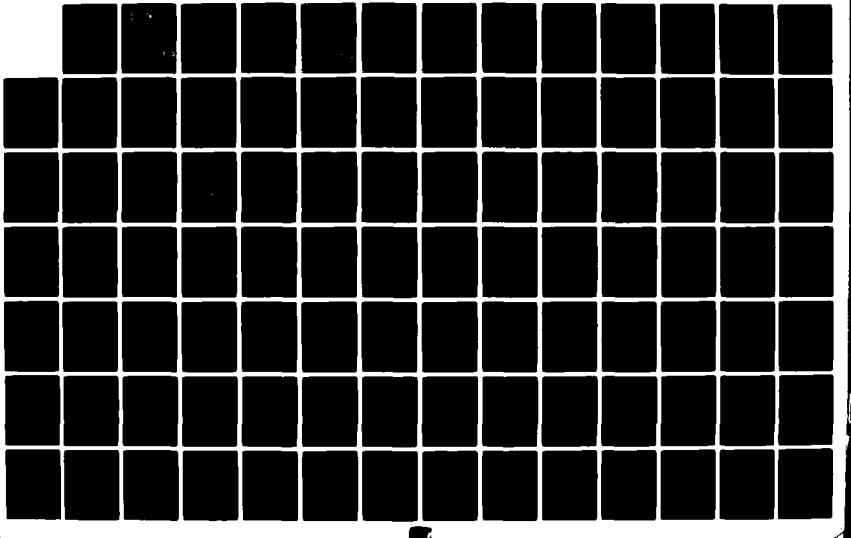
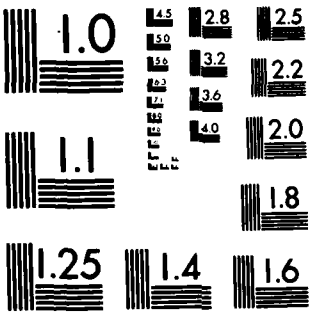
AD-A124 824 DEVELOPMENT OF A DESIGN AUTOMATION GRAPHICS WORK STATION(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING D E SCOTT DEC 82 1/2

UNCLASSIFIED AFIT/GE/EE/82D-60 F/G 9/2 NL






MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A


# DEVELOPMENT OF A DESIGN AUTOMATION GRAPHICS WORK STATION

THESIS

AFIT/GE/EE/82D-60 Donna E. Scott
Capt USAF



DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY (ATC)
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| REPORT DOCUMENTATION PAGE | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|---|---|
| 1. REPORT NUMBER<br>AFIT/GE/EE/82D-60 | 2. GOVT ACCESSION NO.<br>AD-A124824 | 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle)<br>Development of a Design Automation Graphics Work Station | | 5. TYPE OF REPORT & PERIOD COVERED<br>MS THESIS<br>6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s)<br>Donna E. Scott<br>Capt USAF | | 8. CONTRACT OR GRANT NUMBER(s) |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS<br>Air Force Institute of Technology (AFIT/EN)<br>Wright-Patterson AFB, OH 45433 | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS |
| 11. CONTROLLING OFFICE NAME AND ADDRESS | | 12. REPORT DATE<br>Dec 1982<br>13. NUMBER OF PAGES<br>92 |
| 14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) | | 15. SECURITY CLASS. (of this report)<br>UNCLASSIFIED<br>15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Approved for public release: IAW AFR 190-17.

LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (AIC)
Wright-Patterson AFB OH 45433

4 JAN 1983

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

graphics
device-independence
Data Automation (DA)
microprocessors
Motorola 6800

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes the development of a graphics work station which is intended to function as an Input/Output (I/O) processor in support of the Data Automation effort at the Air Force Institute of Technology (AFIT).

This study focused on the requirement definition and design of the graphics work station. Five I/O devices were acquired during this effort and were to be controlled by an M6800

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

microprocessor. A graphics generator was acquired to produce, store, and refresh the graphics display, freeing the microprocessor to collect and process other data.

The hardware and software designs were completed and the hardware which was acquired was interfaced to the M6800. The present configuration of the M6800 EXORciser system prohibited the implementation of the graphics software, however, the device drivers were implemented and tested during this effort. Test Software was generated and used to demonstrate the features of the system that are outlined in this report.

AFIT/GE/EE/82D-60

# DEVELOPMENT OF A DESIGN AUTOMATION GRAPHICS WORK STATION

THESIS

AFIT/GE/EE/82D-60 Donna E. Scott
Capt USAF

AFIT/GE/EE/82D-60

# DEVELOPMENT OF A DESIGN AUTOMATION GRAPHICS WORK STATION

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

by

Donna E. Scott, B.S.E.
Capt USAF
Graduate Electrical Engineering

December 1982

## Preface

This research describes the design of a Data Automation graphics work station. The requirements for this workstation were developed as a part of this effort and the design is based on these requirements. This report documents the requirements definition and design of the graphics workstation and the interface of some Input/Output devices to the controlling microprocessor.

I would like to thank my thesis advisor, Lt Col Harold Carter, for his assistance and encouragement throughout the course of this study. I would also like to thank my readers, Lt Milne and Prof Richard, for their invaluable comments and aid during this study. The excellent support of the laboratory personnel was greatly appreciated. I wish to thank Mr. Dan Zambon, Capt Lee Baker, Mr. Orville Wright, and Mr. Bob Durham for their assistance.

## Contents

List of Figures

## List of Tables

## Abstract

This report describes the development of a graphics work station which is intended to function as an Input/Output (I/O) processor in support of the Data Automation effort at the Air Force Institute of Technology (AFIT).

This study focused on the requirements definition and design of the graphics workstation. Five I/O devices were acquired during this effort and were to be controlled by an M6800 microprocessor. A graphics generator was acquired to produce, store, and refresh the graphics display, freeing the microprocessor to collect and process other data.

The hardware and software designs were completed and the hardware which was acquired was interfaced to the M6800. The present configuration of the M6800 EXORciser system prohibited the implementation of the graphics software, however, the device drivers were implemented and tested during this effort. Test software was generated and used to demonstrate the features of the system.

# I. Introduction

## Background

As digital system complexities continue to increase, manual design, analysis, and fabrication methods have proved inadequate. Design Automation (DA) is the application of computers (with applicable software) to support designers or technicians in the design, fabrication, testing, and maintenance of a digital system. DA can be used to support AF organizations in consulting and evaluating; and the Air Force Institute of Technology (AFIT) is capable of enlarging the currently limited technological field of DA as required by AF and DoD needs (Ref 1).

AFIT also needs DA tools to assist thesis hardware projects, new lab equipment development, and class projects. In addition to hardware design, many theses and class projects involve the design of major software products. Automated aids to support the software engineering of these projects are sorely needed (Ref 1).

The overall stucture of the DA effort at AFIT is identified in Figure 1. Several things will be needed to support the DA research at AFIT. These include:

1. A public-domain database to be used for all DA development
2. A dedicated computer system
3. Significant Input/Output (I/O) capabilities
4. Applications Software

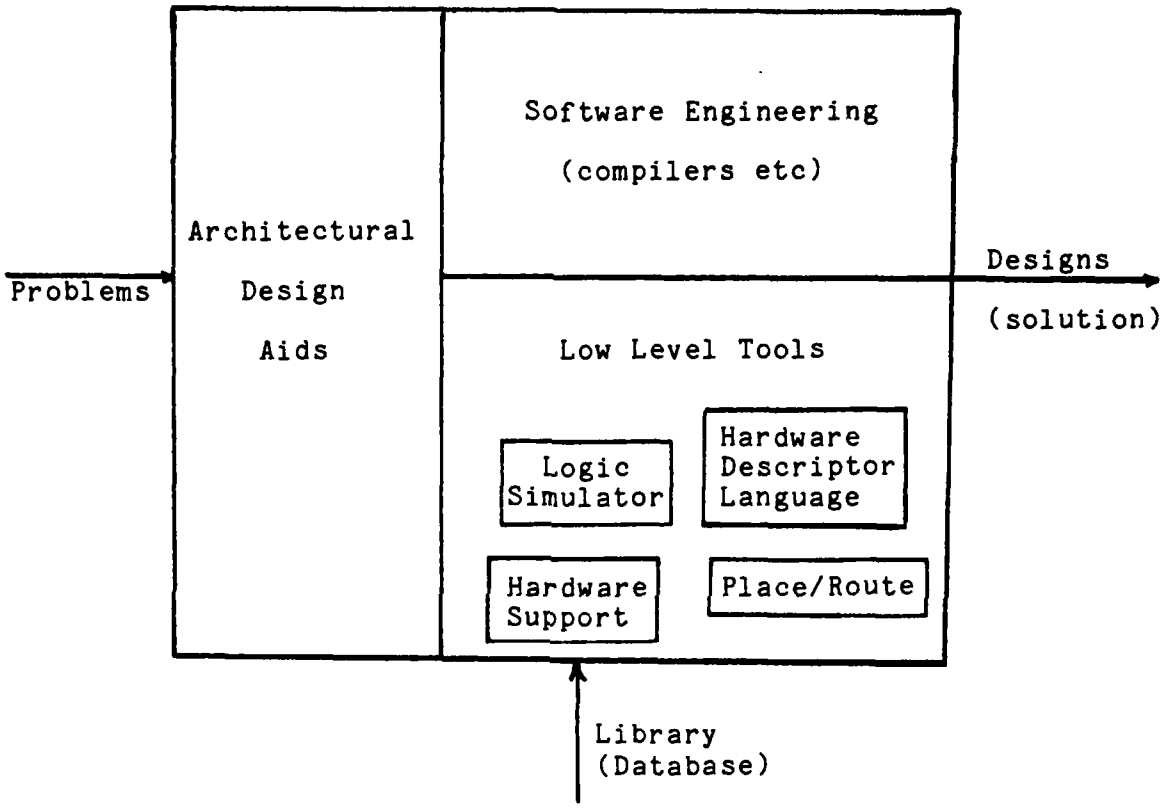



Figure 1. Block Diagram of DA Effort at AFIT

After the hardware and data base have been established, then the integrated development of DA development and software will be carried out.

## Problem and Scope

The purpose of this study was to define, design, and partially implement a graphics work station to support the DA research at AFIT. This work station will be used as an interactive interface between the user and a computer system.

This effort involved the assessment and update of a Motorola M6800 microcomputer system required to control the graphics equipment. The graphics equipment was then interfaced to the M6800.

During the interface phase, the following software was designed:

1. Executive Program
2. Drivers for all I/O devices
3. Graphics subroutines
4. Test and evaluation software

Time precluded the development of application programs.

## Approach

The approach taken was to evaluate the configuration of the M6800 to determine if the graphics equipment interface and software were feasible. This was accomplished during the system orientation phase and the changes required were implemented prior to the interface phase. Definition of the

database of graphics subroutines began during the orientation phase.

A printer and terminal were interfaced first to the M6800 to provide system support during the rest of the hardware buildup. A graphics generator and CRT were interfaced next to establish the basic graphics capability. A tablet and plotter were then interfaced in turn to provide interactive capability and hardcopy graphics.

After all the I/O devices were interfaced and the drivers completed, the system operation was verified. Some programs used to evaluate the graphics generator and the display were taken from the 1351A Graphics Generator Operating and Programming Manual (Ref 2).

Overview

The structure of this report parallels the sequence given in the previous section. Chapter II presents the requirements and standards followed during the design phase. Chapter III details the overall design of the system. Chapters IV and V discuss the hardware and software designs. Chapter VI presents the conclusions and recommendations of the study as well as the lessons learned. Appendix A contains a guide for the user of the EXORciser and Appendix B describes the operation of the graphics work station. Appendix C lists the signals required for the equipment and the cables used to interface it to the system. Appendices D

and E outline the operation of the Quad Comm Module and the serial I/O ports. Appendices F and G contain the details of the system software routines and the listings of the device dependent routines that were implemented.

## II. Requirements and Standards

### System Requirements

The graphics work station is required to support the DA needs at AFIT. It must be capable of obtaining commands from the user, processing them, and presenting a soft copy display. To minimize the fatigue on the operator, response on the graphics CRT should be fast, i.e., maximum delay should not exceed about 10 sec. Delay is defined as the time between the interpretation of the command and the completion of the display on the tube. The display itself must be refreshed at least 25 times a second to minimize flicker.

The graphical presentations must be available in hard copy form. Since the hard copy output is primarily for check plotting, not for final prints, the resolution need not be extremely accurate.

Anticipated applications of the work station include printed circuit board layout, circuit drawing, component positioning, integrated circuit (IC) masking, wire routing, and other two-dimensional diagrams. Three-dimensional displays are not required.

The displays such as the IC masks presentations require multiple overlays. If different colors are not available, varying intensity levels and/or different line types are required for differentiating information on the display.

Because DA requires large data files, the graphics software must include windowing and segmenting functions. Two dimensional transformation functions are also required to perform rotation translations, and scaling of points, segments, and files.

Finally, an interface is required to provide a closed loop between the I/O devices and the user. This interface is the operating system for the graphics workstation. Like any other operating system, it shields the programmer from details of the graphics hardware. It's input subsystem passes user-supplied data to the application program and its output routines handle the actual picture plotting.

Resources

In addition to the microprocessor system equipment (microprocessor, display terminal, disk drive, and printer), the following graphics equipment was used to implement the graphics workstation:

1. Graphics CRT
2. Graphics-Interface Unit
3. Tablet
4. Plotter

The graphics CRT is a 19" monochromatic vector display (HP1310A) which provides the high writing speed, fast settling time, and brightness and contrast needed for the display of high density graphics information. The graphics-interface unit is a graphics generator (HP1251A). Vectors and/or characters are generated and the display is

continually refreshed by the interface-unit, freeing the microprocessor to collect or process other data. The tablet (HIPAD Digitizing Pad) is used as an input device that speeds and simplifies human interaction with the microprocessor. The plotter (HIPLOT DMP-7) provides a simple method of obtaining permanent records of data displayed on the graphics CRT.

The assessment of the M6800 revealed insufficient memory. There were also too few I/O ports to implement the graphics work station. Consequently a full 64K byks of memory and 4 additional serial I/O ports were acquired to support the software and equipment interface.

Standards

The following standards were followed or taken into consideration during this study:

1. Hardware Interface - All the hardwared interfaces were serial interfaces. The RS-232C standard developed by the Electrical Industry Association was used (Ref 3).

2. Software

a. The graphics subroutines were designed so that the applications programs written for this system could take advantage of the Core System Graphics Standard that was developed by the Graphics Standard Planning Committee (GSPC) of the ACM Special Interest Group on Graphics (SIGGRAPH) (Ref 4,5).

The Core System is designed as a general purpose subroutine package that provides an interface between an

application program and graphics hardware, such as line drawing plotters and interactive displays. The principal goal of the Core System is to achieve device independence and thereby to provide portability of graphics programs. To do this, the programmer should strive to localize device dependencies within a few well-defined subroutines in the applications program (Ref 6).

The Core System can be characterized according to five basic areas: output primitives, segmentation, attributes, interaction, and viewing (Ref 5). This thesis effort involved the definition of the subroutines reuired to implement the functions of these five areas.

b. Midwest Scientific Instruments, Inc (MSI) BASIC was used to test some graphics routines. The device driver routines and the other graphics routines were written in M6800 Assembly language.

c. The Motorola Disk Operating System (MDOS) was acquired during this effort, but was not implemented due to the large amount of time required to modify the existing software. The present MSI Disk Operating System, Version 0.985, was used.

# III. System Design

## Overview

The graphics work station is designed as an applications independent system that accepts commands from the user, processes them, and displays them on an output device. The system can be used in a batch mode or interactively. In the batch mode, the user loads and executes a graphics file. In the interactive mode, the user creates or modifies the file using the system's input devices and then executes the file.

All the I/O devices are interfaced to the M6800 EXORciser system. Because of the availability of another host computer to perform data transformations and manipulations, the M6800 will act as an I/O processor. A user's guide for the EXORciser is included as Appendix A.

The interface between the user and the graphics work station is the Executive Program (EXEC). It allows the user to create, modify, and display pictoral data.

The following sections detail the EXEC and discuss the implementation of this user interface. The hardware and software designs are discussed in Chapters IV and V.

## User Interface

The user interface, EXEC, is the operating system for the graphics work station. This section outlines the overall

operation of the EXEC and describes the EDIT, OUTPUT, and INTERPRETER subprograms.

Operation. The program features of the EXEC are presented in Figure 2. They include:

1. LOAD - Reads a program into RAM
2. PURGE - Erases a file
3. RENAME - Changes a filename
4. EDIT - Creates/modifies a file
5. OUTPUT - Displays (executes) a file
6. TRANSFORM - Translates, Rotates, Scales a file
7. HELP - Provides operational information

The first three functions are available with the system's operating system. The others are presented later in this section. A command interpreter is required to decode these instructions from the user. The instructions may be input from the terminal or tablet.

The graphics file contains a list of instructions to be executed by the OUTPUT program. These instructions resemble assembly language instructions, i.e. MOV x,y. A file interpreter is used to assemble each of the instructions into 1 to 5 bytes of machine code. The number of bytes depends on the particular instruction. The coding of the first byte is sufficient to identify the instruction. These instructions and their machine codes are discussed in Appendix B.

After the system initialization process, the EXEC will display a sign-on message such as:

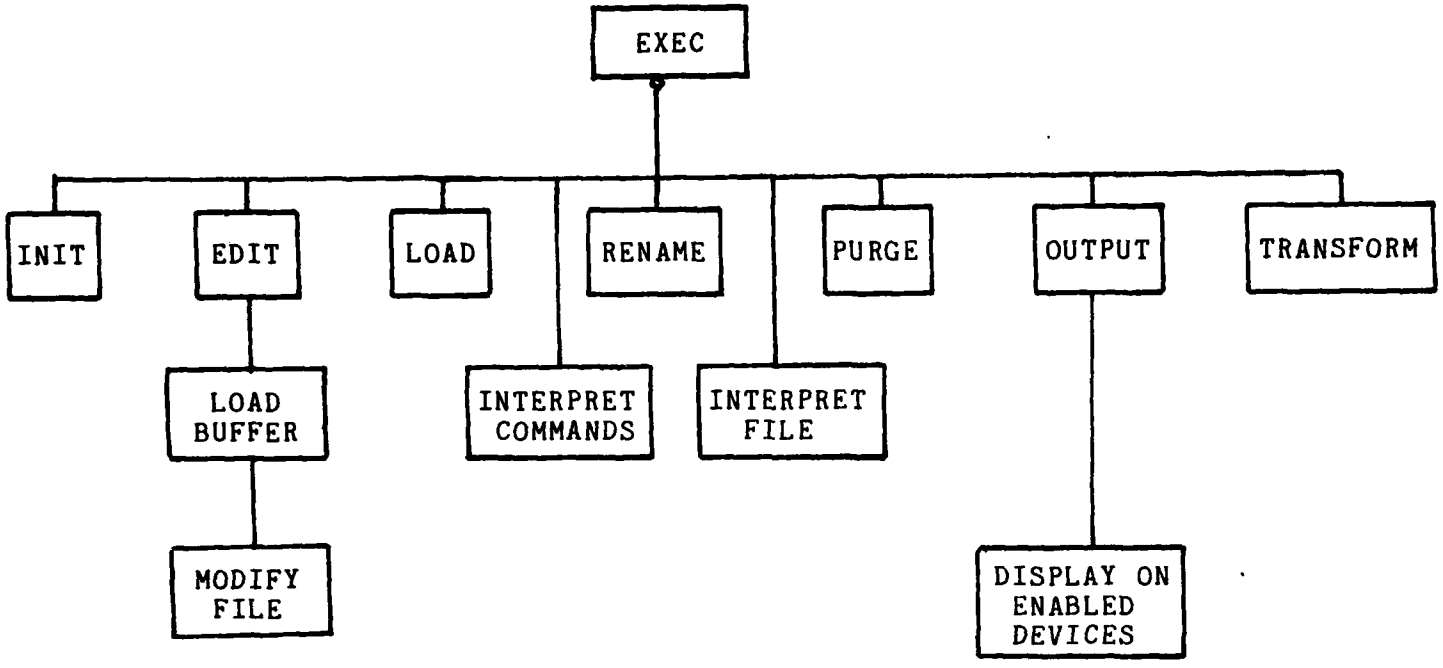



Figure 2. Structure Diagram of User Interface

READY
=

meaning that no errors occurred during the initialization process. In addition, an equal sign (=) is displayed as a prompt indicating that the EXEC is ready to accept commands from the user. The equal sign is subsequently displayed each time the command interpreter gets control. The sign-on message is only displayed when EXEC is reloaded.

EXEC Editor. The success or failure of an interactive program is judged at least as much on its ease of use as on its functional capability (Ref 6). The EXEC Editor allows the user to interactively create a graphics file. Because this program does interact with the user, when implemented, it should:

1. Provide simple, consistent interaction sequences.
2. Prompt the user at each stage of the interaction.
3. Give appropriate feedback to the user
4. Allow graceful recovery from user mistakes.

Messages are used as reminders to the user. Detailed instructions should be included in the EXECs HELP program.

The decomposition of the editor is shown in Figure 3. It has the usual features designed to insert, change, move, delete, and copy lines. Its additional features allow the user to change the input device, delete lines of code related to a particular object, transform the coordinates of an instruction or object and select an output mode that

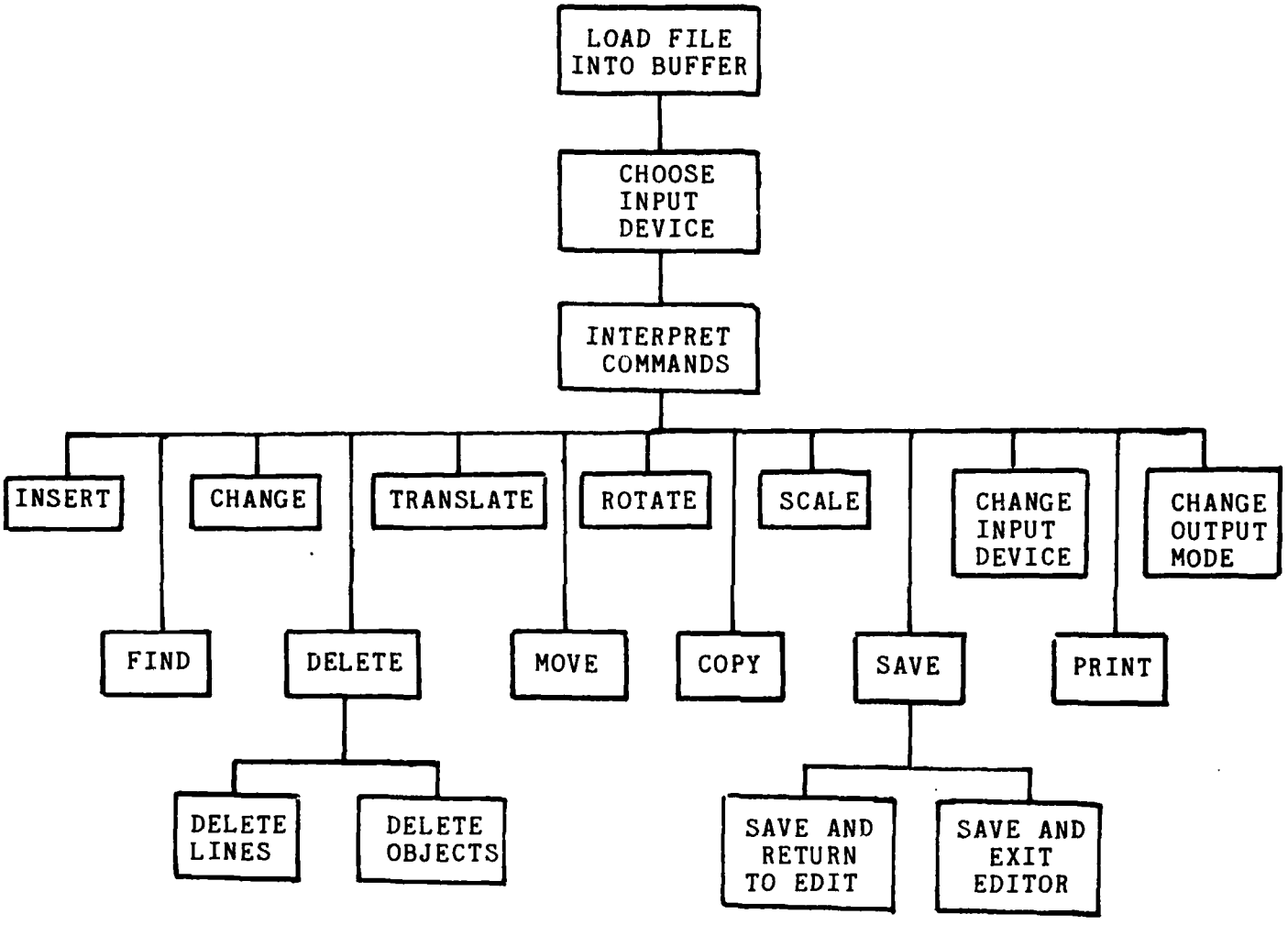



Figure 3. Decomposition of the EXEC Editor

provides an immediate soft copy display of the instructions as they are entered.

EXEC Output. The OUTPUT routine is used to display the graphics file. The graphics file must be an executable file containing the machine code for the instructions and the data. The graphics file is converted to the executable file by the Graphics Interpreter.

This OUTPUT routine is decomposed in Figure 4. It is designed to display a graphics files on the devices that have been enabled in the device enable bit, DEB. The DEB is passed to the OUTPUT routine as an argument. It can be changed dynamically by inserting an OUT command in the graphics files. An example DEB for the devices presently interfaced to the system would assign:

1. Bit Ø - monochromatic graphics CRT
2. Bit 1 - digital plotter
3. Bit 2 - printer

The opcodes and data bytes are sent to the output routines for all the enabled devices. These routines format the commands required to execute the instruction and send them to the devices. In the immediate output mode, each instruction is sent to the graphics generator and is displayed instantly on the graphics display. In the usual batch output mode, if the display is enabled, it is blanked until all the instructions are executed so that the file is displayed instantaneously.

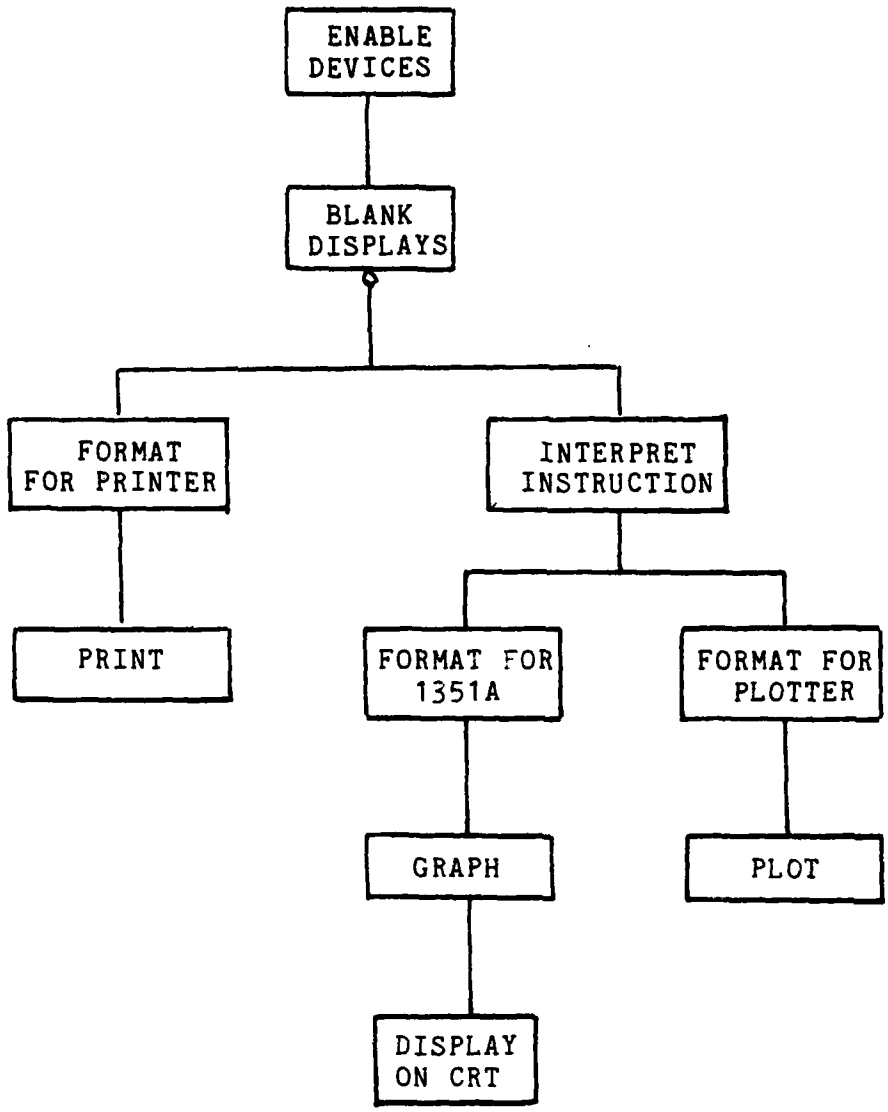



Figure 4. Decomposition of EXECs OUTPUT

Graphics Interpreter. The graphics interpreter is used to convert a graphics file to an executable file. As each instruction is interpreted it is assembled into 1 to 5 bytes of machine code. The first byte is sufficient to decode the instruction. The remaining bytes are data bytes. The instruction byte (opcode) and the data bytes are used by the OUTPUT routine to display the file on the output devices.

## Suggested Implementation

The EXEC and its file manipulation routines should be written in a high level language. The subroutines should be placed in a system library so that they may be accessed by any aplications program. If the user has a detailed knowledge of the system, it would be very useful to have access to these routines. Since the EXORciser system does not support high level languages or linking in its present configuration, it should be reconfigured with the Motorola Disk Operating System before attempting to implement the software required for the graphics work station.

# IV. Hardware Design

## Overview

The graphics work station's hardware consists of an M6800 based microcomputer system, the EXORciser, with several I/O devices interfaced (see Figure 5). In addition to the standard I/O devices-disk drive, display terminal, and line printer - the graphics equipment includes a graphics generator, graphics CRT, digitizing pad, and digital plotter.

The graphics equipment was interfaced to the EXORciser during this effort. The driver routines were produced to verify the communicatin between the M6800 and the devices. Test software was used to verify the operation of the equipment.

All the graphics equipment interfaces were designed as asynchronous, serial, RS-232C level interfaces. This chapter describes the graphics equipment, their interfaces, and the applicable serial I/O port configurations.

## Equipment Description

Graphics Generator. The HP 1351A graphics generator converts digital inputs to analog outputs capable of driving large screen computer graphics displays. It receives digital information from the RD-232C interface bus for storage in its internal memory. Vectors and/or characters

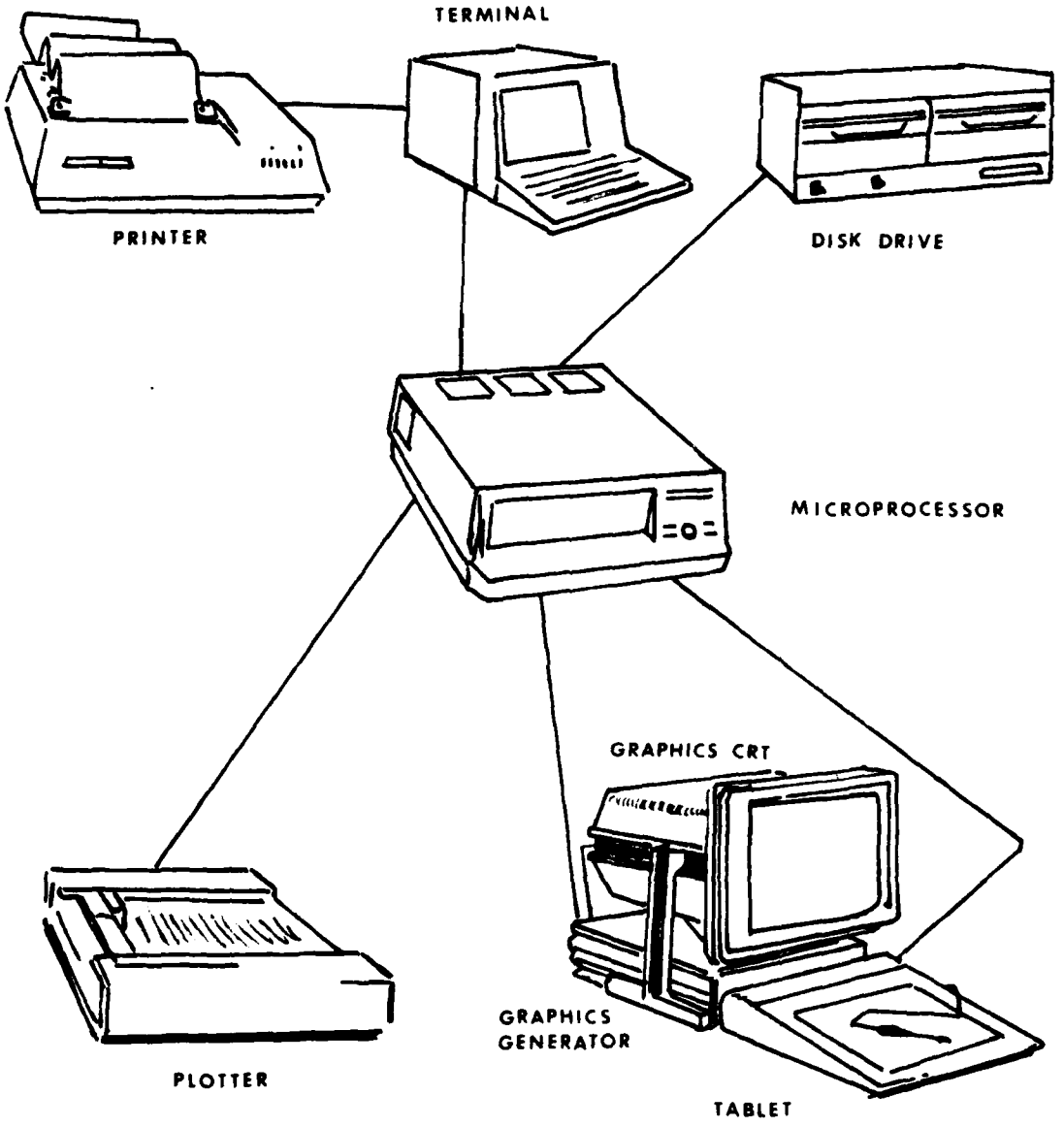



Fig 5 Graphics Work Station Equipment

are generated, and the display is continually refreshed by the 1351A, freeing the EXORciser to collect or process other data. The graphics generator can address and individually display 1020 x 1020 points on the CRT display. The graphics display is vector oriented and alphanumerics can be displayed.

The 1351A has 64 memory files which are selectable in size, separately addressable and eraseable. Variable vector drawing speeds provide three intensity levels for highlighting selected information.

Each digital word in the 1351A can be a vector coordinate or an ASCII character. Each character can be programmed to be displayed in four different sizes and can be displayed normally or else rotated 90 degrees counterclockwise.

Graphics CRT Display. The HP13110A is a 19" monochromatic, vector display which offers the high writing speed, fast settling time, and the brightness and contrast needed for the display of high density graphics information. This large screen display has an X-Y bandwidth response up to 5 MHz. Deflection sensitivity is one volt for full scale deflection on either axis. Settling time, defined as the time required for the spot to settle to within one spot diameter of its final position, is less than 1 us.

The screen resolution of the 1310A is 50 lines/inch in the 11 x 11 inch quality area. The spot resolution is 0.020 inches. The screen's writing speed is greater than 10 in/us

and has rise times for both X and Y amplifiers less than 75 ns. This means that the 1310A can draw vectors at a rate of over one million inches per second. This is useful in Computer Aided Design (CAD) applications which require complex, high density drawing capability (Ref 7).

Digitizing Pad. The HI PAD digitizing pad (tablet) is a compact 11 x 11 inch active surface digitizer with all of its electronics built into a single surface. It has the capability of digitizing through non-metallic materials up to approximately 1/8 inch in thickness.

The tablet has a basic resolution of 100 lines/inch. It can be increased to 200 lines/inch. Its data rate is up to 100 coordinate pairs per second.

There are two sensing elements that can be used to determine the coordinates on the active surface of the tablet: the cursor and the stylus. The cursor has a pushbutton which is referred to as the cursor button. This button allows the user to select a point mode where one coordinate is digitized each time the button is pressed or the Switch mode. In the switch mode, digitizing occurs continuously when the cursor button is pushed.

The tablet is normally used with the stylus. The stylus operates just like an ordinary ball-point pen and is used with the tablet in a stream mode. In this mode, digitizing occurs continuously. The proper position of the stylus is shown by the illumination of the tablet's RESET switch/indicator.

Digital Plotter The HIPLOT DMP-7 digital plotter is a microprocessor-based plotter that uses Digital Microprocessor Language (DM/PL). This simplifies the task of creating a graph by placing sophisticated functions within the Z-80 processor itself. The DM/PL may be used with BASIC, PASCAL, FORTRAN, or any other high level languages. Simple letters are used to denote moving and drawing functions in the plotter. Nine different line types are available with DM/PL to enhance the readability in more complex plots. The symbol set contains 93 printable ASCII characters which may be drawn in five sizes and rotated clockwise, 0 degrees, 90 degrees, 180 degrees, or 270 degrees clockwise.

The transmitted block size is limited to 256 bytes of data between "handshakes". After the data string (which ends in a prompt code) is sent to the plotter, it is stored in the plotter's input data buffer. An input operation is initiated to assist the plotter's handshake response. When there is sufficient room in the buffer for more data, the plotter sends a <CR> to the host computer to let it know that it is ready for the next data string.

## Equipment Interface

This section outlines the interface of the graphics equipment as well as the interface of the serial ports required by the equipment.

All interfaces to the EXORciser were designed as

asynchronous serial, RS-232C level interfaces. Since none of the equipment used RS-232C standard signals on all interface pins, special interconnect cables were designed for each piece of equipment. The connector signals and interconnect cables are outlined in Appendix C. The data transmission configurations of the equipment are listed in Table 1.

Graphics Generator. The graphics generator was shipped from the factory in an RS-232C modem configuration. Consequently, its serial port was configured as a terminal to interface it to the M6800. This configuration is shown in Figure 14 of Appendix D. The input and output signals are listed in Table 4 and its interconnect cable is shown in Figure 8, both of Appendix C. Clear-to-Send and Request-to-Send flags control the flow of data into the graphics generator.

Graphics CRT. The graphics CRT connects directly to the graphics generator via three (3) BNC connectors. These provide the X,Y, and Z analog voltage signals required to drive the display.

TABLET. The tablet is not configured as a modem or terminal, so its serial port configuration was not specified. Since the other serial ports were configured as terminals, the tablet's serial port was also configured as a terminal. Table 5 and Figure 9 of Appendix C outline the connectors signals and the interconnect cable. There are no signals from the tablet to control the flow of data out of

Table 1

Graphics Equipment Data Transmission Configuration

| | Graphics Generator | Tablet | Plotter |
|---|---|---|---|
| no. start bits | 1 | 1 | 1 |
| no. data bits | 8 | 8 | 8 |
| no. stop bits | 2 | 2 | 1 |
| parity | none | none | none |
| band rate | 9600 | 4800 | 9600 |
| connector signals | Table 4 | Table 5 | Table 6 |
| interconnect cable | Figure 8 | Figure 9 | Figure 10 |

the tablet. A user-defined buffer must be set-up prior to calling the TABLET input routine. All data will be transferred to the buffer as it is read from the tablet. The user can then manipulate the data as required. Any tablet switch functions must be implemented in the software.

Plotter. The plotter is configured as an RS-232C modem. Its serial port was configured as a terminal to interface it to the M6800. DM/PL protocol controls the flow of data to the plotter. Its signals and interconnect cable are shown n Table 6 and Figure 10 of Appendix C.

Serial Port Interface

In order to support the interface of the graphics equipment to the EXORciser, an M68MM07 Quad Communications Module (Quad Comm Module) was interfaced to the M6800. This module provides four serial I/O ports known as Asynchronous Communications Interface Adapters (ACIAs). This module interconnects directly to the EXORciser bus. The operation of the Quad Comm Module and the signals provided by the bus are discussed in Appendix D.

Module Base Address. All four ACIAs on this module share the same base address. The four states of address bits A1 and A2 select a given ACIA. The board was delivered with a base address of $EC20. (A '$' indicates a hexadecimal number.) The memory map of the EXORciser was examined to determine if this base address had to be changed. The memory map is shown in Figure 6 and the

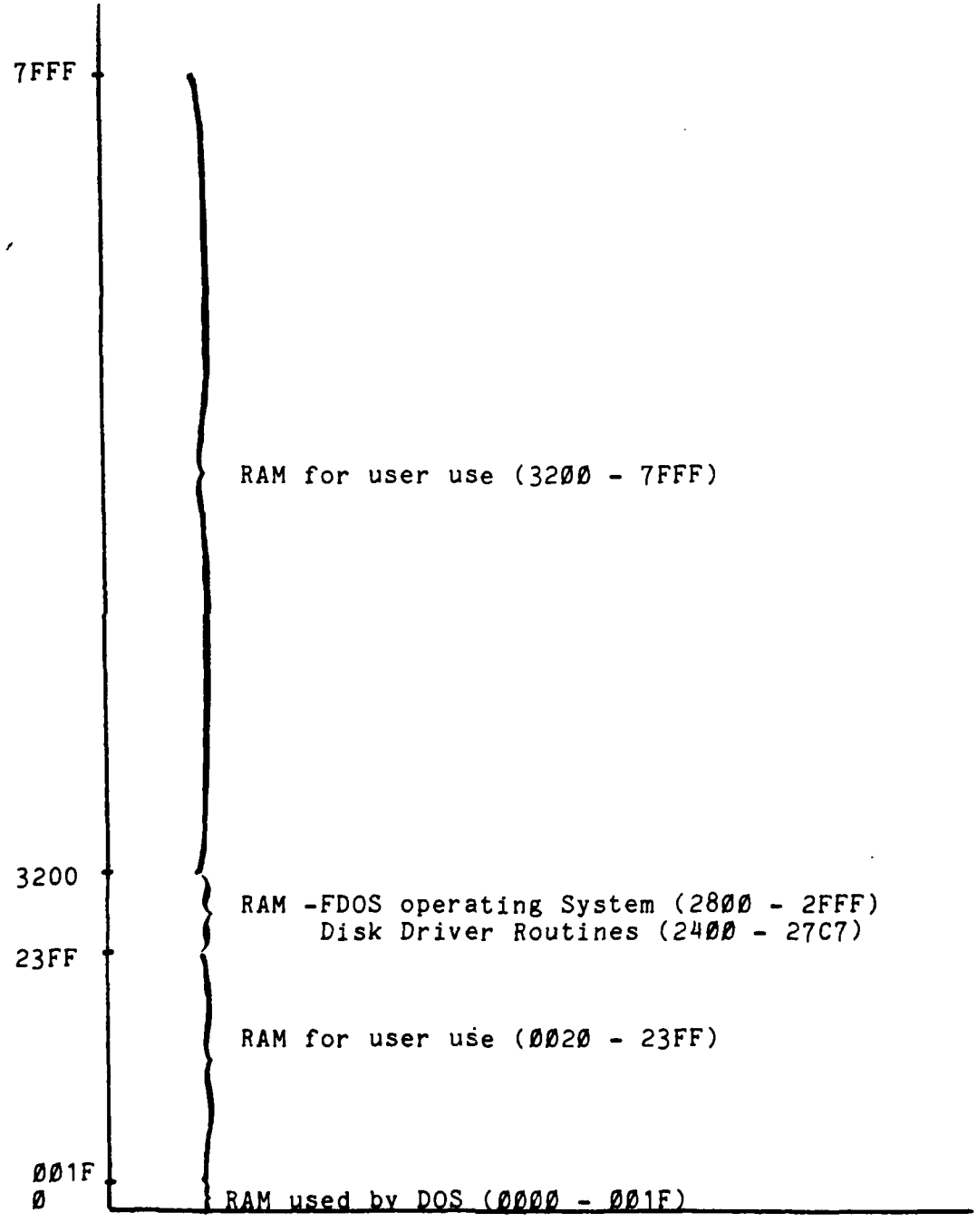



Figure 6. EXORciser Memory Map (1 of 2)

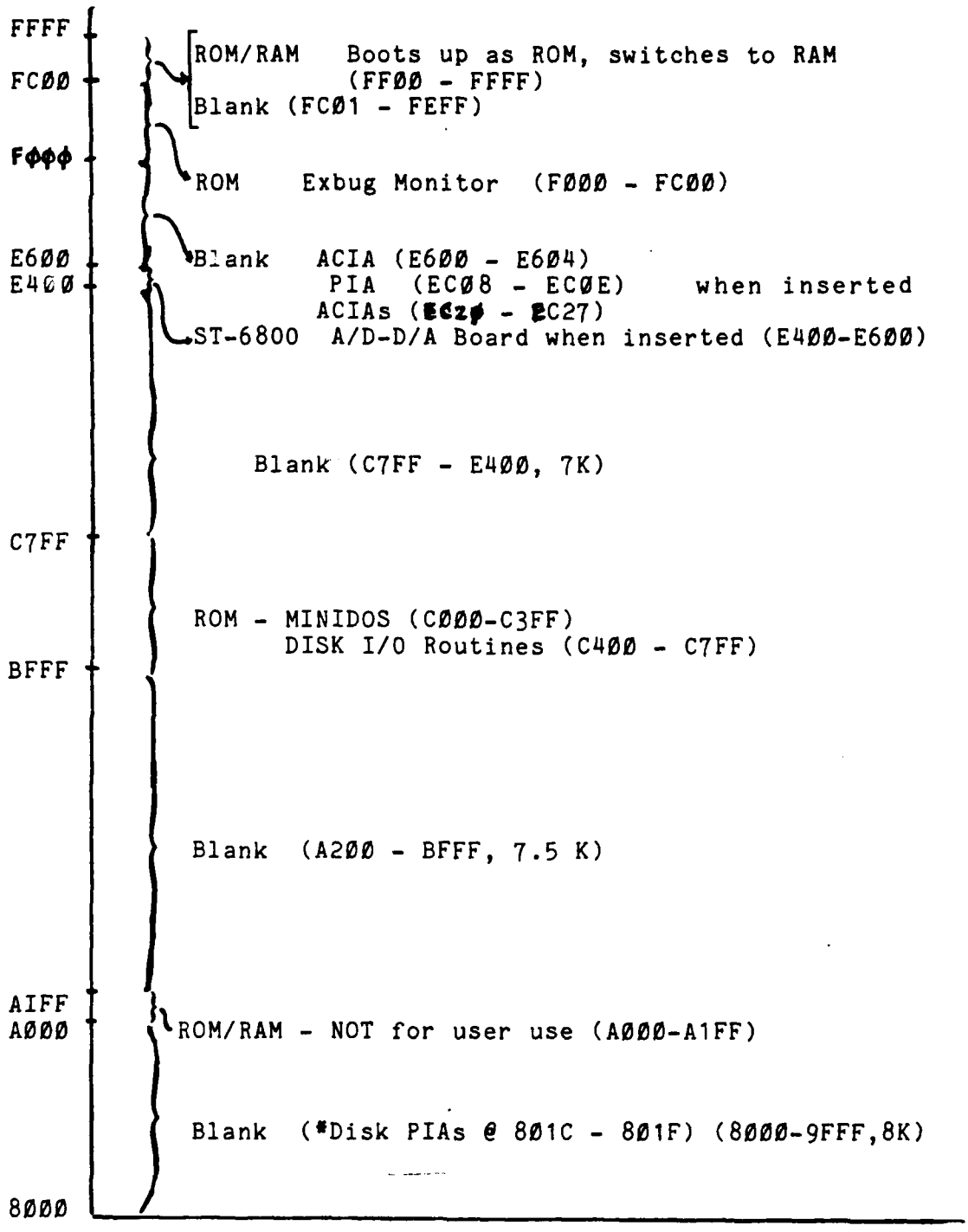



Figure 6. EXORciser Memory Map (2 of 2)

reasons for not changing the base address are discussed below.

Memory organization is very fragmented with the user areas located below $8000. Because of the time that would be required to reorganize the memory and modify the software that would be affected, the memory was not reorganized. I/O ports presently used by the system are located at $E600 and $EC09, so a base address of $EC20 places the new serial ports in an area not available to the user and in an area of memory with other I/O ports. This location in memory is compatible with DOS and MDOS operating systems. For these reasons, the base address of the Quad Comm Module was not changed.

Serial Port Assignments. From the M6800s addressing point of view, each ACIA is simply two memory locations that are treated in the same manner as any other read/write memory. The operation of the ACIA is discussed in Appendix E

The four ACIAs of the Quad Comm Module are assigned as follows:

1. ACIA1 ($EC20,$EC21) - Graphics Generator
2. ACIA2 (#EC22,$EC23) - Tablet
3. ACIA3 ($EC24,$EC25) - Plotter
4. ACIA4 ($EC26,$EC27) - Auxiliary

Summary

The interface of the equipment outlined in this chapter provides the capability for soft and hard copy displays.

The soft copy display can be refreshed as fast as 260 times every second so that flicker is minimized.

For more information on the operation of the equipment, refer to the Appendices and the Operators manuals.

# V. Software Design

## Intro

The design of the graphics system's software affects the maintenance, update, and use of the system. This chapter is concerned with the design of the software component of the system.

The software for the graphics work station is designed so that any applications program can be run with only a knowledge of the inputs required for the subroutines and their functions. The details are invisible to the user, i.e., any formatting required by the graphics equipment is performed within those routines.

The software has been divided into the following categories:

1. Device Managers - Driver routines for the graphics equipment
2. Graphics Package - Routines used to generate pictures and to handle graphical interaction.
3. Minipulation Package - Routines that operate on a graphics file.

The software hierarchy is shown in Figure 7. The EXEC accesses the routines in the graphics package as well as the file manipulation package. The design and organization of the software categories are discussed in the following sections. The internal design of the functions and subroutines is presented in Appendix F. The listings of the device

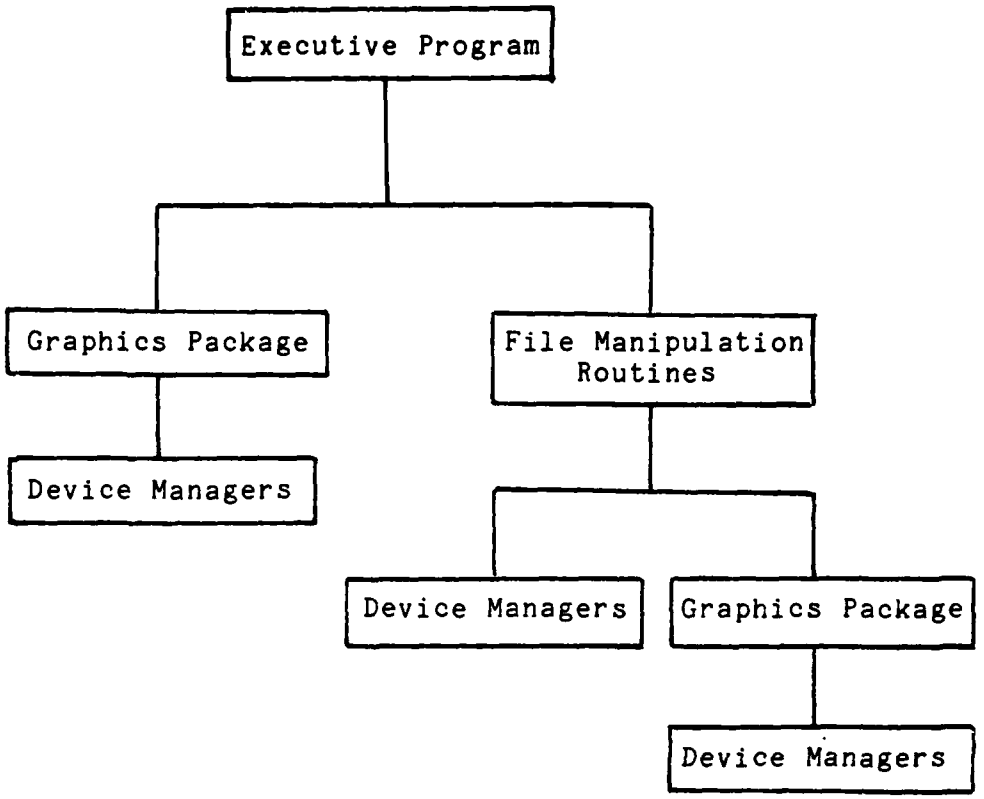



Figure 7. System Software Organization

dependent routines that were implemented are included as Appendix G.

Device Managers

The device routines are hardware dependent. They include the driver routines for the graphics generator, tablet, and plotter.

The system software is designed so that the graphics routines and applications programs access the upper level of the device managers - the command transmitters. These transmitters transfer and receive a string - one or more data bytes. They are used to transfer and receive commands from the I/O devices. These transmitter routines are the only ones that can access the driver routines.

Initialization of the serial ports and graphics generator is performed by a routine that is called once at the start of the graphics session. Since the plotter requires initialization after every 256 bytes transferred, its initialization is performed by the routine that formats commands for the plotter. The tablet does not require any type of initialization.

Graphics Package

In designing a graphics package, consideration must be given to the issues that affect the usefulness of the package (Ref 8). Some of the important issues considered in designing this package are:

1. Simplicity - Features that are too complex for the aplication programmer to easily use and understand will not be used.

2. Consistency - A consistent graphics system is one that behaves in a predictable manner. Function names, calling sequences and error handling should follow simple and consistent patterns. Experience shows that warnings in the user's manual of system anomolies are often overlooked by the reader.

3. Robustness - The graphics system may be misused by the applications programmer. Whether this is through misunderstanding or through mischievous enjoyment of trying everything once, the system should be able to accept this treatment. A serious error should be reported in the most helpful manner possible. Only in extreme circumstance should errors cause termination of execution since this could cause the user to lose valuable results.

4. Completeness - There should be no ommissions in the set of functions provided by the system; missing functions will have to be provided by the user who may not have the knowledge or access to the computers resources to be able to write them.

Tne graphics package is not designed for any specialized application requirements. Although it does not provide every imaginable graphics facility, it does provide a small set of functions that can conviently handle a wide range of applications.

The functions in the graphics package can be divided into sets, each set concerned with a particular kind of task. During the design and testing of the routines, it is easier to check for completeness if each set of routines is concerned with only one function.

There are a number of function sets (Ref 8). The ones chosen for this graphics package are:

1. Graphics primitives - These are used to display straight lines, text strings, circular arcs, and other simple graphical items.

2. Windowing functions - These allow the programmer to define the boundary of the visible portion of the picture.

3. Segmenting functions - These routines allow the user to divide the graphics file into segments. They allow easier modification to the picture.

4. Two-dimensional transformatin functions - These are used for translating, rotating, and scaling the display.

5. I/O functions - These allow the user to input commands with a keyboard or graphical input device and output these commands to the output devices.

6. Utility functions - These are the miscellaneous functions that may be required by any of the other routines.

File Manipulation Package

These routines provide the overall capabilities of the

graphics workstation. They allow the user to create, modify, display, plot, erase, rename, and transform a data file. They can be accessed by any applications program and are used by the EXEC.

These routines are outlined in Chapter III. They and the other routines described in this chapter are discussed in more detail in Appendix F.

Summary

The software outlined in this chapter provides the features required by a graphics work station. It allows the user to input commands to produce a graphical display. it also allows the user to input, modify, and manipulate graphics files.

The graphics software is device independent and provides functions that can conviently handle a wide range of applications. It provides the functions required to manipulate the large data files rquired in Data Automation applications.

# VI. Conclusions and Recommendations

## Conclusions

The graphics workstation outlined in this thesis provides a set of functions that can be used for a wide range of applications. The system is designed so that aplications programs can be prepared with only a knowledge of the inputs required for the system's software. The user need not understand the particulars of the I/O devices handled by the system.

The system has five I/O devices: printer, graphics CRT, digital plotter, display terminal, and tablet. These devices were interfaced to the EXORciser during this effort. The software was designed for the system, however, the EXORciser is not configured to support its implementation because of the limited memory, lack of linking capability and lack of high level languages. If the graphics system is implemented as designed, it should be able to provide the graphics capabilities required to support the Data Automation effort at AFIT.

## Recommendations

The Motorola M6800 based EXORciser system is not currently configured correctly to support the large, complex computer programs required to operate the graphics work station. The EXORciser's memory should be increased and

reorganized to maximize the contiguous memory available to the user and to centralize the memory used by the system's I/O devices. The disk interface should be redesigned and implemented to provide more reliable disk accesses. The present Disk Operating System should be replaced by the Motorola Disk Operating System to allow the addition of a Linker/Loader and high level languages to support the implementation of the graphics system's software.

## Lessons Learned

The recommendations for the EXORciser reconfiguration were developed during the system orientation and graphics database definition phases of this effort. Four problem areas were identified:

1. Documentation
2. Operating System
3. Memory
4. Disk Interface

Each of these areas will be discussed in the sections below.

Documentation. The documentation for the operation of the M6800 EXORciser system was extensive. However, the documentation available on creating, assembling, and executing an assembly language program on this particular system was insufficient. The test programs that were written during the orientation phase were completed mainly through trial and error. The procedures and idiosyncrasies of the system were documented in a user's guide. This information is stored with the system documentation.

Operating System. The present operating system supports only a small and insufficient set of resident functions. The only high level language available is BASIC. A particular frustration occurs because groups of subroutines (which must be divided into several programs because of memory) cannot be linked together during assembly. All the programs must have the exact location in memory of the others. This makes relocatable code and system libraries of subroutines impossible. The Motorola Disk Operating System has been acquired and should be implemented to replace the present operating system, MSI DOS.

Memory. The memory map of the EXORciser is included in this report as Figure 6. It shows that the memory is fragmented. The additional memory acquired, but not implemented as a part of this effort, should be interfaced to the system. The I/O areas of memory and that used by the operating system should be consolidated so as to maximize the amount of contiguous memory available to the user.

Disk Interface. An H27 disk interface was designed and interfaced prior to this effort. Many problems associated with this interface surfaced during this study. The features of the operating system were inconsistently available to the user, i.e. editing, purging, and assembling a program, packing the disk, and copying files from one disk

to another. This interface should be reevaluated and redesigned or rebuilt as necessary.

The problems that exist in the current system make it impossible to implement the graphics work station. These changes should be implemented so that the system can be used as an I/O processor for the graphics work station as well as a system that permits M6800 software development.

## Bibliography


1. Carter, Harold, W., Professor of Electrical Engineering. A Plan for Digital Systems Design Automation at AFIT. Unpublished Paper. School of Engineering, AFIT, WPAFB, OH, Nov 1981.

2. Hewlette-Packard Co. Model 1351A Graphics Generator Operating and Programming Manual. Colorado Springs, Colorado: Hewlett-Packard Co, October 1981.

3. McNamara, John E. Technical Aspects of Data Communication. United States: Digital Press, 1977.

4. Michener and van Dam. "Recent Efforts Toward Graphics Standardization," ACM Computer Surveys, 10(4): 365-380 (Dec 78).

5. Michener and van Dam. "A Functional Overview of the Core System with Glossary," ACM Computer Surveys, 10(4): 381-388 (Dec 78).

6. Bergeron, Bono, Foley, "Graphics Programming Using the Core System," Computing Surveys 10(4): 389-443 (Dec 78).

7. Potts, J., "Computer Graphics - Whence and Hence," Computers and Graphics,1: 137-156 (1975).

8. Newman, William and Robert Sproull. Principles of Interactive Graphics. United States: McGraw-Hill, 1979.

9. Warner, James and Nikolaus Kiefhaber, "Implementing Standard Device-Independent Graphics," Mini-Micro Systems, (Jul 82).

10. Motorola Inc. M68MM07 Quad Communications Module Micromodule 7. Motorola Inc., 1979.

11. Motorola Inc. M6800 EXORciser User's Guide. Motorola Inc., 1975.

12. Moore, A.W. et al. Microprocessor Applications Manual. New York: McGraw-Hill Book Company, 1975.

## Appendix A

## User's Guide to the EXORciser System

This appendix describes the procedure for booting the EXORciser and running a simple program. The guidelines for writing a program are outlined in a User's guide that is stored with the system's documentation.

The program to be run is designed to accept commands from the user and display them on the graphics CRT. These commands must be formatted for the HP1351A graphics generator. The formats are outlined in the Operating and Programming Manual for the 1351A (Ref 2). The program is stored as two separate routines. Each of these routines must be loaded and executed. Since the RUN command of the system does not function properly, the execution procedure is as follows:

1. Load the program from disk
2. Return control to the monitor (EXBUG)
3. Execute the program starting at a given memory location from MAID

The two routines required for this display program are:

1. TRANS - initializes the equipment and transfers a string of ASCII characters to the graphics generator.

2. INPUT - accepts the commands from the terminal and sends them to the graphics generator using TRANS.

The commands required to run this input program and the system's response to the inputs are listed below. The underlined text is what is typed by the user after resetting the EXORciser:

```
:9
EXBUG1.1 MAID
*E200:G
DOS READY
LOAD TRANS

DOS READY
EXBUG
:9
EXBUG 1.1 MAID
*0100:G

ACIA INITIALIZED

1351A INITIALIZED

DOS READY
LOAD INPUT

DOS READY
EXBUG
:9
EXBUG 1.1 MAID
*3200:G

GRAPHICS COMMANDS MAY BE ENTERED FROM THE KEYBOARD
EACH COMMAND MUST BE TERMINATED BY A ':'
```

At this point, the user types a command. Once a ':' is typed, the system responds with

IS THIS CORRECT?

The command is sent to the graphics generator only if the response is Y (yes). The following message is printed next:

DO YOU WANT TO CONTINUE (Y OR N)?

If the answer is yes, the program will accept another command from the terminal If not, the system signs off with:

```
IT'S BEEN NICE TALKING TO YOU
DOS READY
```

If the program is accidently terminated, the following commands will restart it:

```
EXBUG
:9
EXBUG 1.1 MAID
*3200;G
```

The intro message will be repeated and commands can be input.

A sample input is printed below. If these commands are typed after the input message has been printed (or after the 1351A memory has been erased), a star will be displayed on the CRT.

```
PE0,;
IS THIS CORRECT?
Y
DO YOU WISH TO CONTINUE (Y OR N) ?
PA50,50;;
IS THIS CORRECT?
Y
DO YOU WISH TO CONTINUE (Y OR N) ?
Y
PE1,;
IS THIS CORRECT?
Y
DO YOU WISH TO CONTINUE (Y OR N) ?
Y
PA175,300;300,50;20,200;330,200;50,50;;
IS THIS CORRECT?
Y
```

This program was used to verify communication between the EXORciser and the graphics generator and between the graphics generator and the graphics CRT. It is written in assembly lanugage and is included in the User's guide. It can be updated to provide a more sophisticated input routine if desired.

## Appendix B

## Graphics Work station I/O Description

This appendix outlines the input and output procedures of the graphics work station. There are presently two input devices and three output devices. The input devices are the display terminal and the tablet. The output devices are the printer, plotter, and graphics CRT.

### Input

The user inputs a set of instructions from the input devices. These instructions are listed in Table 2 and are used to build a graphics file which resembles an assembly language program. The terminal inputs are input by an editor routine. The TABLET routine reads the coordinates from the tablet. The switch functions (instructions) and data bytes are then placed in the graphics file.

An example of code which moves the cursor to (5Ø,5Ø) and then draws a circle of radius 1Ø is given below:

```
MOV 5Ø,5Ø
CIR 1Ø
```

These instructions will be assembled into 1 - 5 bytes of machine code by the file interpreter. The number of bytes depends on the instruction. The first byte of the machine code is sufficient to decode the instruction. The remaining bytes are data bytes. The instruction byte

Table 2

Graphic File Instructions

| Inst | Subroutine | Inst | Subroutine |
|---|---|---|---|
| NOP | No operation | MKR | Marker (n) |
| PNT | Point | CRE | Createsegment (name) |
| MOV | Move (x,y) | OSG | Opensegment (name) |
| LIN | Line (x,y) | CSG | Closesegment |
| TXT | Text ('TEXT') | DSG | Deletesegment (name) |
| CIR | Circle (r) | SCA | Scale (x,y) |
| TYP | Type (n) | ROT | Rotate (θ) |
| SZE | Size (n) | TRA | Translate (x,y) |
| ORI | Orientation (n) | OUT | Output (DEB) |
| COL | Color (n) | CLR | Clearscreen |
| INT | Intensity (n) | WIN | Window (xmin, xmax, ymin, ymax) |
| CIR | Circle (r) | VPT | Viewport (xmin, xmax, ymin, ymax) |

(opcode) and the data bytes are passed as arguments to the subroutines required to implement then.

The hexadecimal equivalents of the binary codes are listed in Table 3. There are 25 defined machine codes, 231 of the possible 256 being unassigned. This allows room for growth for the graphics workstation.

Output

The OUTPUT routine of the EXECUTIVE program executes the machine code of the graphics file. Any or all of the output devices may be selected to display the file. Each instruction is then sent to the output subroutines for the enabled devices.

If the graphics CRT is an output device, OUTPUT sends the instruction to GRAPH, the routine that formats instructions for the HP1351A graphics generator. The graphics generator then drives the graphics CRT. For the plotter, the PLOT routine formats the instructions required to drive the plotter. If the printer is selected, the mnemonics in the graphics file are listed.

There are two types of output modes, continuous and update. In the continuous mode, as each instruction is inserted into the graphics file, the display is updated to reflect the change. The graphics CRT would normally be the only output device in this mode. For the update mode, the display is updated only on request. Any of the devices could be used in this mode.

Table 3

Hexadecimal Values of Machine Codes

| OPCODE | NMEUMONIC | OPCODE | NMEUMONIC |
|---|---|---|---|
| 00 | NOP | | |
| 01 | * | 21 | CRE |
| 02 | PNT | 22 | CSG |
| 03 | MOV | 23 | DSG |
| 04 | LIN | 24 | OSG |
| 05 | TYP | 25 | * |
| 06 | TXT | 26 | SCA |
| 07 | SZE | 27 | ROT |
| 08 | ROT | 28 | TRA |
| 09 | COL | 29 | * |
| 0A | MKR | 2A | * |
| 0B | CIR | 2B | * |
| 0C | * | 2C | * |
| 0D | * | 2D | * |
| 0E | * | 2E | * |
| 0F | * | 2F | * |
| 10 | OUT | 30 | CLR |
| 11 | * | 31 | WIN |
| 12 | * | 32 | VPT |
| 13 | * | 33 | * |
| 14 | * | 34 | * |
| 15 | * | 35 | * |
| 16 | * | 36 | * |
| 17 | * | 37 | * |
| 18 | * | 38 | * |
| 19 | * | 39 | * |
| 1A | * | 3A | * |
| 1B | * | 3B | * |
| 1C | * | 3C | * |
| 1D | * | 3D | * |
| 1E | * | 3E | * |
| 1F | * | 3F | * |
| 20 | * | . | . |
| | | . | . |
| | | FF | * |

## Future Considerations

There are presently five I/O devices for the graphics work station. Any number of input devices can be added to the system as long as the required interface software is implemented. The Device Enable Byte (DEB) is used to enable the output devices. A total of 8 can be interfaced and only 3 are presently designed for the system. If more than 5 devices must be added to the system, the DEB would have to be expanded.

## Appendix C

## Interconnect Cables for the Graphics Equipment

This appendix outlines the input and output signals of the graphics generator, tablet and plotter. The signals required to interface the equipment to the M6800 are shown in the interconnect cable diagrams. Each diagram lists the ACIA signals and pin numbers on the left. The particular piece of equipment's signals and pin numbers are on the right.

Each cable was constructed using a 20-pin edge connector on the Quad Comm Module (at the ACIA) and 25-pin "D" connector to the equipment.

Table 4

HP-1351A Interface Connector Signals

| Pin | Direction | Name and/or Function |
|---|---|---|
| 1 | - | Protective ground, shield |
| 2 | in | Serial ASCII data in |
| 3 | out | Serial ASCII data out |
| 4 | in | Request to Send, resets the 1351A |
| 5 | out | Clear to Send, tells the computer that a transmission can start or must stop |
| 6 | out | Data Set Ready, power is "on" in the 1351A |
| 7 | - | Signal ground, common return |
| 8 | out | Carrier detect, tied to Clear to Send |
| 15 | out | Transmission clock, signal is RS-232 voltage level compatible |
| 16 | out | Transmission clock, clock is TTL Signal level compatible |
| 17 | out | Transmission clock, signal is RS-232 voltage level compatible |
| 20 | - | Undefined, user option |
| 24 | in | Receive clock, signal is RS-232 voltage level compatible |

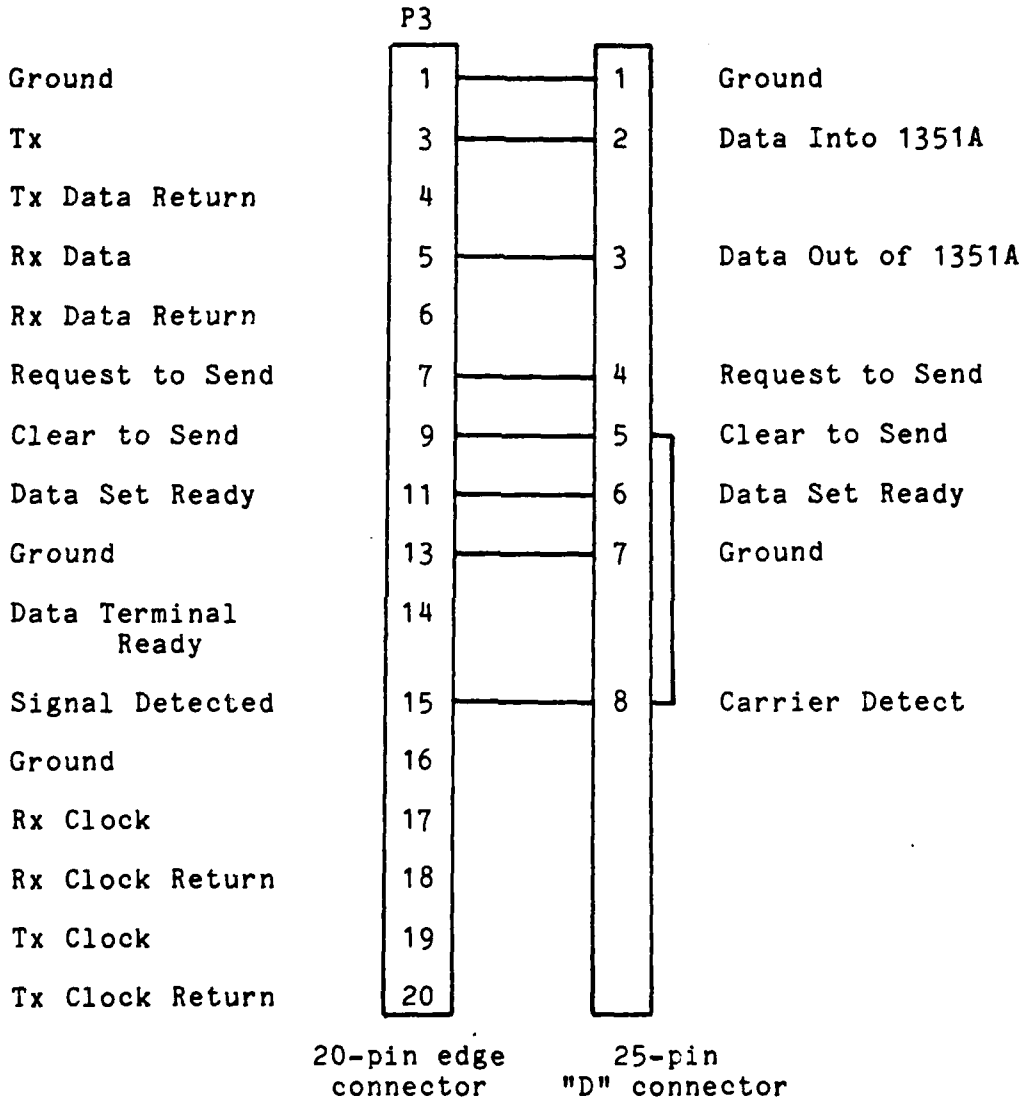



Figure 8. Interconnect Cable for HP1351A

Table 5. Table Interface Connector Signals

| Pin | Name | Comments |
|---|---|---|
| 1 | Data Bit Ø | |
| 2 | Data Bit 1 | |
| 3 | Data Bit 2 | The data bits are used for parallel data formats in conjunction with one or more data strokes |
| 4 | Data Bit 3 | |
| 5 | Data Bit 4 | |
| 6 | Data Bit 5 | |
| 7 | Data Bit 6 | |
| 8 | Data Bit 7 | |
| 9 | Display STRB | |
| 10 | BCD STRB | |
| 11 | Binary STRB | |
| 12 | TTL Serial Out | |
| 13 | ACK | For synchronization, BCD or Binary Format |
| 14 | +12 Volts DC | External Power |
| 15 | Band Rate Select | |
| 16 | Band Rate Select | |
| 17 | IN/MM Select | Strap pin 17 to pin 20 for metric selection |
| 18 | Resolution Select | Connect pin 18 to pin 20 for 0.005 in resolution |
| 19 | Origin Select | Connect pin 19 to pin 20 for fixed origin |
| 20 | Ground | Circuit Common |
| 21 | Reset | Contact closure to ground resets digitizer |
| 22 | RS-232 | Serial Output |
| 23 | Cursor Switch | Duplicates cursor switch function |
| 24 | PROX | Indicates the cursor is in digitizing position |
| 25 | +5 Volts DC | Used to power Optional Display only |

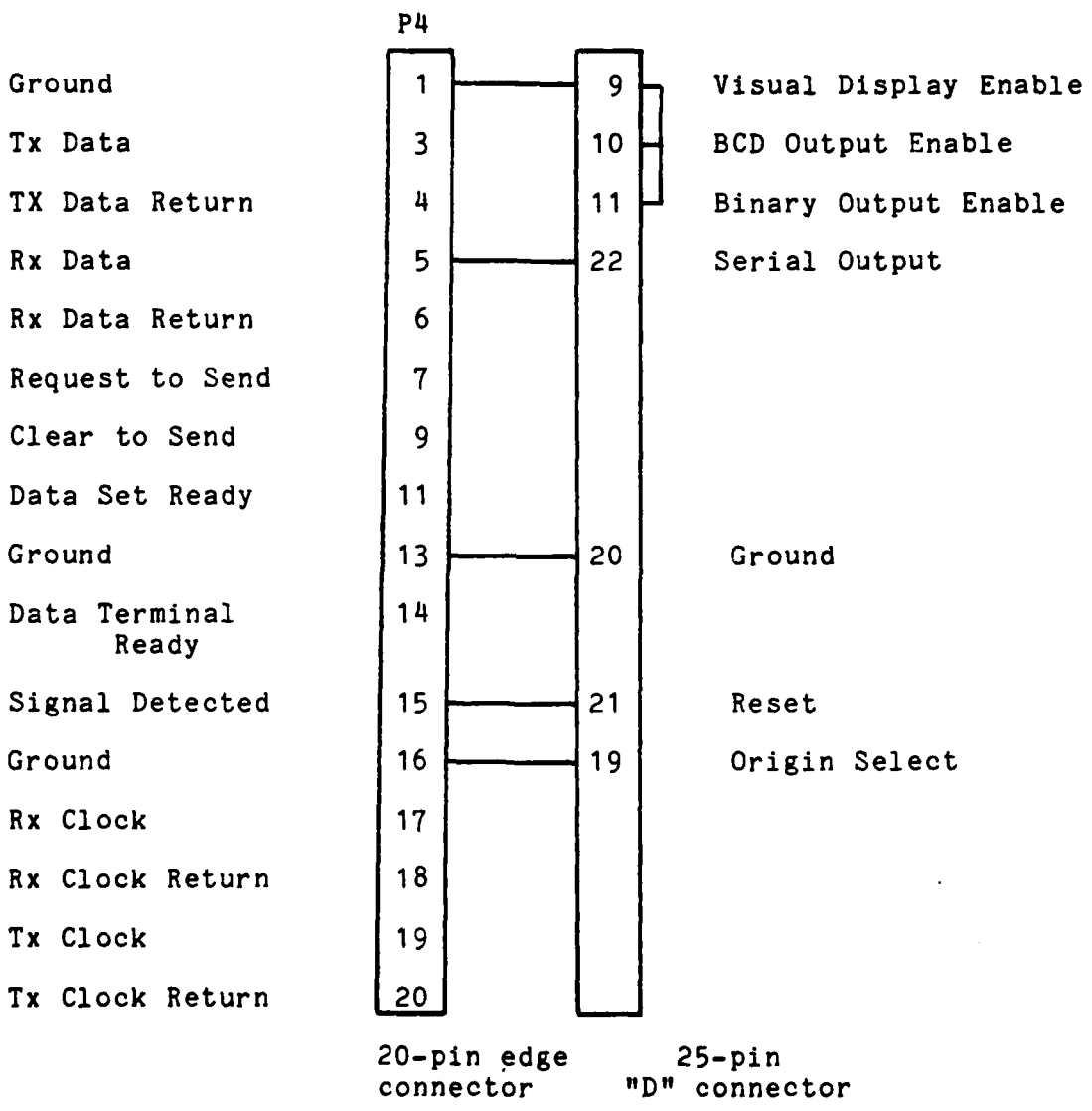



Figure 9. Interconnect Cable for Tablet

Table 6

Plotter Interface Connector Signals

| Pin # | Signal |
|---|---|
| 1 | Ground |
| 2 | Transmitted Data (From Plotter) |
| 3 | Received Data (To Plotter) |
| 4 | Request to Send connected together on plotter |
| 5 | Clear to Send |
| 6 | Band rate clock in |
| 7 | Ground (Signal common) |
| 9 | Hand shake mode select |
| 14 | 9600 |
| 15 | 4800 |
| 16 | 2400 Connect appropriate pin to pin 6 at |
| 17 | 1200 the plotter end of the I/O cable |
| 18 | 600 to select band rate. |
| 19 | 300 |
| 20 | READY |

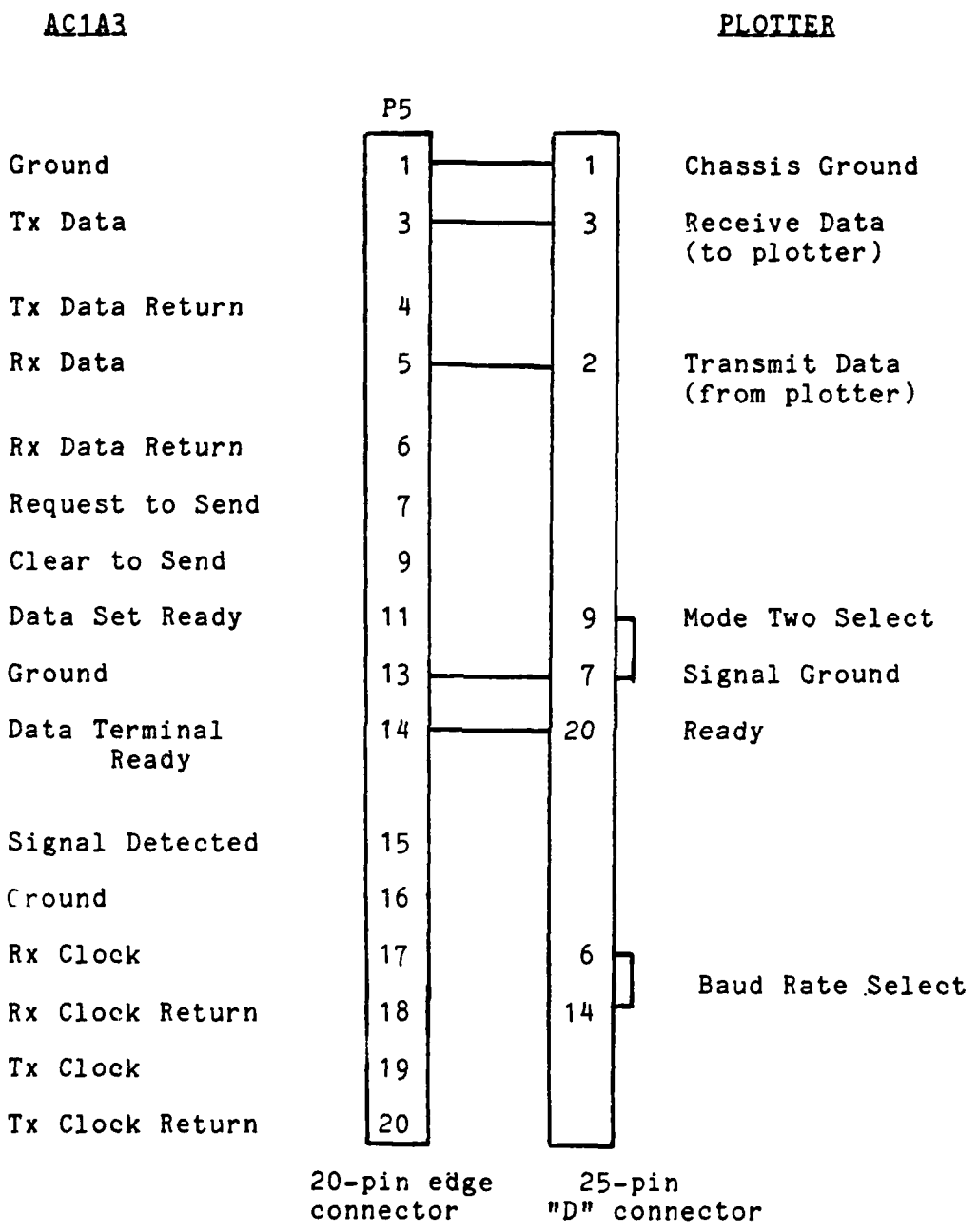



Figure 10. Interconnect Cable for Plotter

## Appendix D

## Quad Communications Module

### Overview

The M68MM07 Quad Communications Module (Quad Comm Module) provides four serial I/O channels. Each channel may be configured to appear as a data terminal or as a modem. The configuration of each channel on the module is entirely independent of the others.

A block diagram of the Quad Comm Module is shown in Figure 11 (Ref 10:3-2). It receives the 16 address bits, AØ through A15, along with the timing signals, the Valid User's Address (VUA), and the Read/Write (R/W) command.

The address decoder consists of the address line enable/disable jumpers at K21, transparent latching, coincident decoding, and combination with address bits A1 and A2 uniquely select one of the four ACIAs. Each of the four enable lines is routed to its respective ACIA. The operation of the ACIA is discussed in Appendix E.

The Quad Comm Module interconnects directly with the EXORciser BUS. The signals provided by the BUS are identified in Table 7 (Ref 11).

### Module Address Select

All four I/O channels share the same base address. Address bits A1 and A2 are used to select a given port out

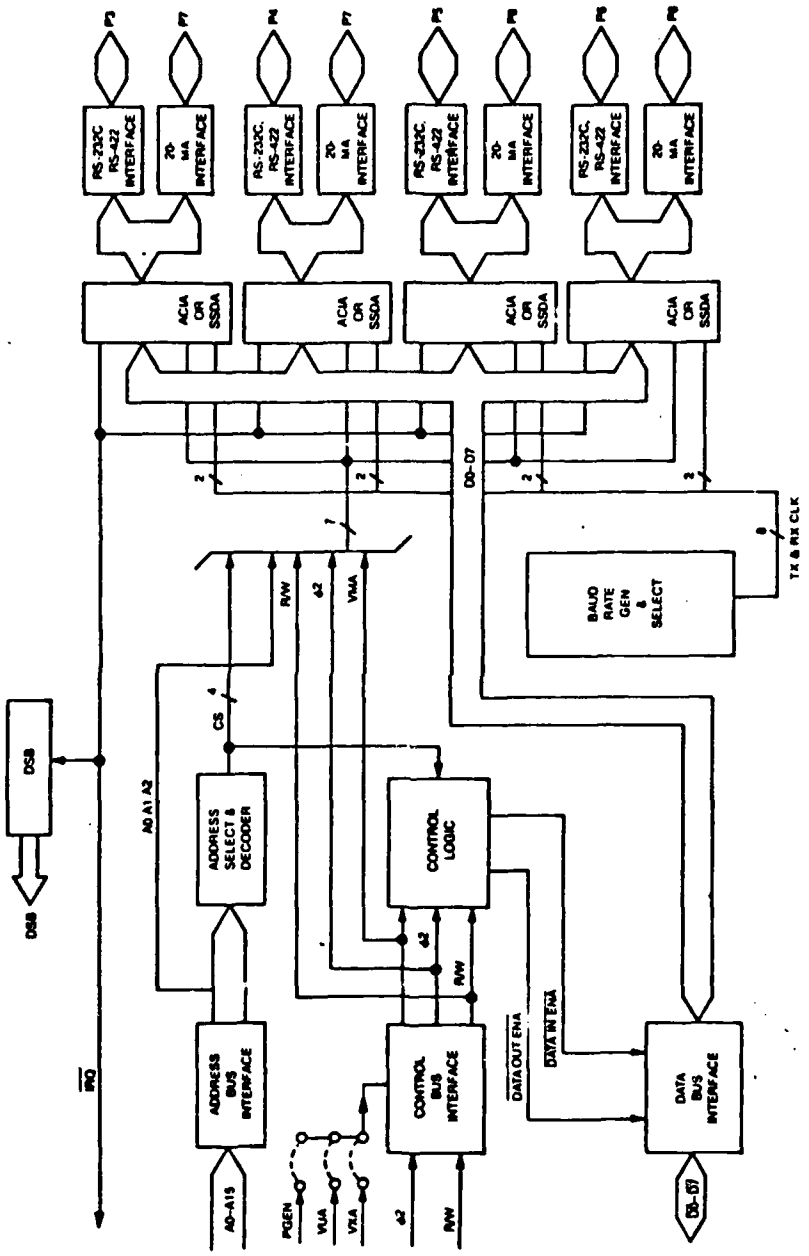



Fig 11 Quad Communications Module Block Diagram (Ref 10:3-2)

Table 7 (Ref 11:2-8)

EXORciser BUS Signals (1 of 6)

| PIN NUMBER | SIGNAL MNEMONIC | SIGNAL NAME AND DESCRIPTION |
|---|---|---|
| A | +5 VDC | +5 VDC used for the module's logic circuits. |
| B | +5 VDC | +5 VDC used for the module's logic circuits. |
| C | +5 VDC | +5 VDC used for the module's logic circuits. |
| D | $\overline{\text{IRQ}}$ | INTERRUPT REQUEST ($\overline{\text{IRQ}}$) - This signal requests that MPU interrupt sequence be generated within the machine. The MPU will wait until it completes the current instruction that it is executing before it recognizes this request. At that time, if the interrupt mask bit in the MPU condition code register is not set (interrupt masked), the MPU will begin the interrupt sequence. |
| E | $\overline{\text{NMI}}$ | NON-MASKABLE INTERRUPT ($\overline{\text{NMI}}$) - This signal requests that a non-maskable interrupt be generated within the machine. The MPU will wait until it completes the current instruction that it is executing before it recognizes the NMI signal. At that time, the MPU will begin its non-maskable interrupt sequence. The EXORciser EXbug Firmware uses the non-maskable interrupt for its Abort, Run-One-Instruction, and Stop-On-Address functions. |
| F | VMA | VALID MEMORY ADDRESS (VMA) - This output indicates to the Debug Module that there is a valid memory address on the address bus. This signal is not three-state. |
| H | | Not used |
| J | Ø | Phase 2 (Ø2) clock signal |
| K | GND | GROUND for ±12 VDC |
| L | MEMCLK | MEMORY CLOCK (MEMCLK) -- This signal is the basic clock frequency used by the MPU Module to generate its Ø1 and Ø2 clock signals. This signal also is used by the dynamic memory modules. |

| | | |
|---|---|---|
| M | -12 VDC | -12 VDC for use with user's custom designed interface circuitry. |
| N | $\overline{TSC}$ | THREE STATE CONTROL ($\overline{TSC}$) - This input, when high, places all of the address lines in their off or high impedance state. The VMA and BA signals will be forced low. The data bus is not affected by the TSC input. |
| P | BA | BUS AVAILABLE (BA) - This bus available output signal will normally be a low level; when activated, it will go high indicating that the MPU has stopped and that the address bus is available. This will occur if the Go-Halt line is in the Halt (low) state or the MPU is in the WAIT state as a result of executing a WAIT instruction. At such time, all the MPU three-state output drivers will go to their off state and other outputs to their normally inactive state. A maskable interrupt, or actuating the RESTART or ABORT switches, removes the MPU from the WAIT state. |
| R | MEMRDY | MEMORY READY (MEMRDY) - This signal enables the EXORciser to work with slow memories. When low, this signal inhibits the EXORciser from generating Ø1 and Ø2 timing signals with Ø2 present. At the completion of its memory operation, the slow memory returns the MEMRDY signal to a high level. |
| S | REFCLK | REFRESH CLOCK (REFCLK) - This signal is generated by the dynamic memory module being used as the master refresh module. This signal is used to initiate a memory operation on the dynamic modules used as slave refresh modules. |
| T | +12 VDC | +12 VDC for use with user's custom designed interface circuitry. |
| U | BAT +12 | BATTERY +12 VOLTS (BAT +12) - This line in normal EXORciser operation is +12 VDC from the EXORciser power supply via the battery backup option. If the EXORciser is using battery backup and the power supply is turned off or a power-fail should occur, the power backup places +12 VDC on this line. If the EXORciser is not using battery backup, this line is open. |

| | | |
|---|---|---|
| V | $\overline{\text{STDBY}}$ | STAND BY ($\overline{\text{STDBY}}$) - This line is a low level during a power-fail condition and a high level during normal EXORciser operation. |
| W | | Not used |
| X | | Not used |
| Y | | Not used |
| Z | | Not used |
| A | | Not used |
| B | GND | GROUND |
| C | | Not used |
| D | | Not used |
| E | (BA) | BUS AVAILABLE (BA) - This line is present only when an Evaluation Module I is placed in the EXORciser chassis. Signal is same as BA on pin P1-P above. |
| F | (G/H) | GO-HALT - This line is used only when an Evaluation Module is placed in the EXORciser chassis. Signal is same as G/H on P1-4. |
| H | D3 | DATA BUS (D3) - This bi-directional line, when enabled, provides a two-way data transfer between the MPU Module and all other plug-in modules in the EXORciser. The data bus drivers on these module are in their off or high-impedances state except when one of these modules is selected during a memory read operation. |
| J | $\overline{\text{D7}}$ | DATA BUS ($\overline{\text{D7}}$) - Same as $\overline{\text{D3}}$ on P1-H. |
| K | $\overline{\text{D2}}$ | DATA BUS ($\overline{\text{D2}}$) - Same as $\overline{\text{D3}}$ on P1-H. |
| L | $\overline{\text{D6}}$ | DATA BUS ($\overline{\text{D6}}$) - Same as $\overline{\text{D3}}$ on P1-H. |
| M | A14 | ADDRESS BUS (A14) - This address line, when enabled, transfers the MPU program counter output to address the plug-in modules in the EXORciser. |
| N | A13 | ADDRESS BUS (A13) - Same as A14 on P1-M. |
| P | A10 | ADDRESS BUS (A10) - Same as A14 on P1-M. |

| | | |
|---|---|---|
| R | A9 | ADDRESS BUS (A9) - Same as A14 on P1-M. |
| S | A6 | ADDRESS BUS (A6) - Same as A14 on P1-M. |
| T | A5 | ADDRESS BUS (A5) - Same as A14 on P1-M. |
| U | A2 | ADDRESS BUS (A2) - Same as A14 on P1-M. |
| V | A1 | ADDRESS BUS (A1) - Same as A14 on P1-M. |
| W | GND | GROUND |
| X | GND | GROUND |
| Y | GND | GROUND |
| 1 | +5 VDC | +5 VDC used for the module's logic circuits. |
| 2 | +5 VDC | +5 VDC used for the module's logic circuits. |
| 3 | +5 VDC | +5 VDC used for the module's logic circuits. |
| 4 | G/H | GO/HALT (G/H) - When this input is in the high state, the MPU will fetch the instruction addressed by the program counter and start execution. When low, all activity in the MPU will be halted. This input is level sensitive. In the halt mode, the MPU will stop at the end of an instruction, bus available will be at a high level, valid memory address will be at low level, and all other three-state lines will be in their off or high impedance state.<br>The halt line must go low with the leading edge of the phase one clock (Ø1) to insure single instruction operation. If the halt line does not go low with the leading edge of the phase one clock, one or two instruction operations may result, depending on when the halt line goes low relative to the phasing of the clock. |
| 5 | RESET | RESET - This signal is used to start the MC6800 MPU and reset the EXORciser plug-in modules whether the MC6800 MPU is performing a memory read (high) or write (low) operation. The normal standby state of this signal is read (high). Also, when the MC6800 is halted, this signal will be in the read state. |
| 7 | Ø1 | Phase 1 (01) clock signal |

| | | |
|---|---|---|
| 8 | GND | GROUND for ±12 VDC |
| 9 | GND | GROUND for ±12 VDC |
| 10 | VUA | VALID USER's ADDRESS (VUA) - This signal, when high, indicates that the address on the address bus is valid and the EXORciser is not addressing its EXbug program. |
| 11 | -12 VDC | -12 VDC for use with the user's custom designed interface circuitry. |
| 12 | $\overline{\text{REFREQ}}$ | REFRESH REQUEST ($\overline{\text{REFREQ}}$) - This signal, when low, initiates a memory refresh cycle of the dynamic memory modules. During the refresh operation the MPU Module is inhibited from generating its 01 and 02 clock signals. |
| 13 | REF GRANT | REFRESH GRANT (REF GRANT) - This signal, when high, instructs the dynamic memory modules to refresh their memories. |
| 14 | | Not used |
| 15 | | Not used |
| 16 | +12 VDC | +12 VDC for use with the user's custom designed interface circuitry. |
| 17 | BAT +12 | BATTERY +12 VOLTS (BAT +12) - Same as BAT +12 on P1-U. |
| 18 | ($\overline{\text{TSC}}$) | Three-State Control ($\overline{\text{TSC}}$) - This line is present only when an Evaluation Module is placed in the EXORciser chassis. Signal is same as TSC on P1-N. |
| 19 | | Not used |
| 20 | | Not used |
| 21 | AC OFF | AC OFF - In a system using battery backup, this signal indicates when ac power is removed from the primary of the power supply transformer. |
| 22 | | Not used |
| 23 | | Not used |
| 24 | GND | GROUND |

| | | |
|---|---|---|
| 25 | | Not used |
| 26 | | Not used |
| 27 | | Not used |
| 28 | | Not used |
| 29 | $\overline{D7}$ | DATA BUS ($\overline{D7}$) - Same as D3 on P1-H |
| 30 | $\overline{D5}$ | DATA BUS ($\overline{D5}$) - Same as D3 on P1-H |
| 31 | $\overline{D0}$ | DATA BUS ($\overline{D0}$) - Same as D3 on P1-H |
| 32 | $\overline{D4}$ | DATA BUS ($\overline{D4}$) - Same as D3 on P1-H |
| 33 | A15 | ADDRESS BUS (A15) - Same as A14 on P1-M. |
| 34 | A12 | ADDRESS BUS (A12) - Same as A14 on P1-M. |
| 35 | A11 | ADDRESS BUS (A11) - Same as A14 on P1-M. |
| 36 | A8 | ADDRESS BUS (A8) - Same as A14 on P1-M. |
| 37 | A7 | ADDRESS BUS (A7) - Same as A14 on P1-M. |
| 38 | A4 | ADDRESS BUS (A4) - Same as A14 on P1-M. |
| 39 | A3 | ADDRESS BUS (A3) - Same as A14 on P1-M. |
| 40 | A0 | ADDRESS BUS (A0) - Same as A14 on P1-M. |
| 41 | GND | GROUND |
| 42 | GND | GROUND |
| 43 | GND | GROUND |

of the four available. The selectable base address range is $PCNM (A '$' indicates a hexadecimal number) where:

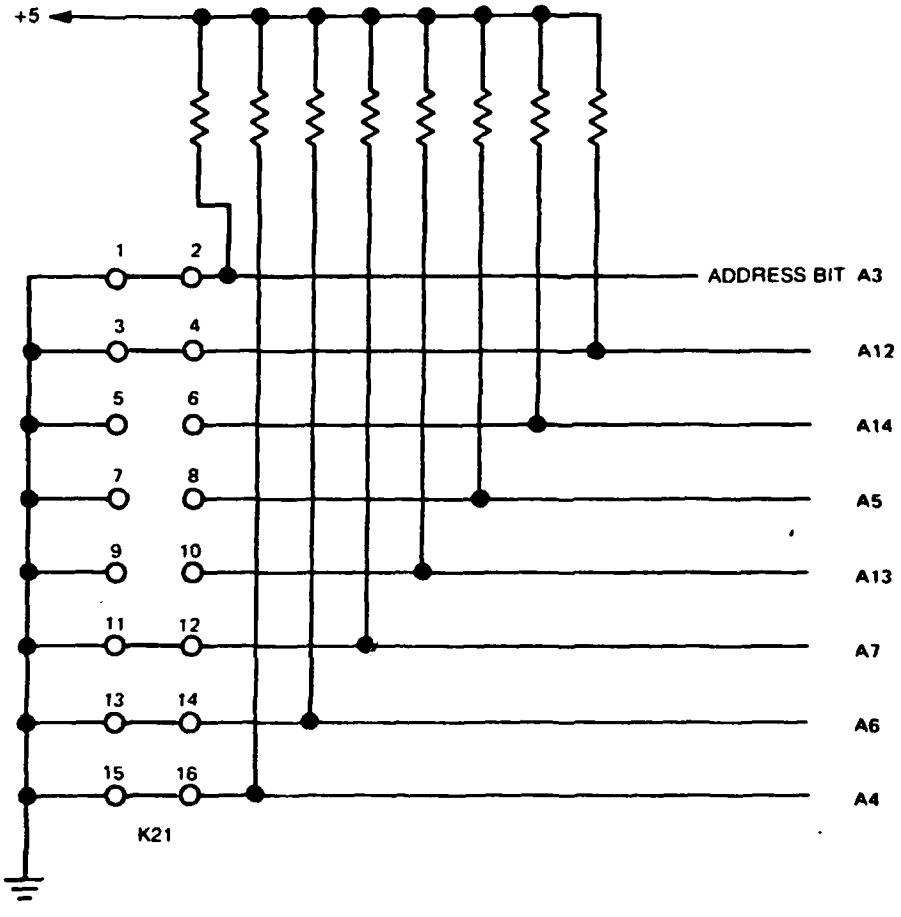
P = $(8 through F)
C = $C
N = $(Ø through F)
M = $(Ø or 8)

Eight address bits - A3, A4, A5, A6, A7, A12, A13, and A14 - are under user control. A shorting jumper in K21 will require the specified address bit to be a logical "zero" - low. Ommission of the shorting jumper will require the specified address bit to be a logical "one" - high (see Figure 12).

The board was delivered with K21 strapped to address $EC2Ø by jumpers at K21 pins 1 and 2 (bit A3), 15 and 16 (bit A4), 13 and 14 (bit A6), 11 and 12 (bit A7), and 3 and 4 (bit A12).

Port Configuration

Each port of the Quad Comm Module can be configured as an RS-232 modem or terminal. To configure a port as an RS-232C modem, jumper wire pin straps K3 through K6 and K12 through K15 as shown in Figure 13. To configure as an RS-232 data terminal, jumper wire pin straphs K3 through K6 and K12 through K15 as shown in Figure 14. For any other configurations refer to the Quad Comm Module documentation (Ref 10).




INSERT A GROUND STRAP FOR EACH ADDRESS LINE REQUIRED TO BE NEGATIVE, (LOGIC ZERO) IN GENERATION OF THE BASE ADDRESS.

Fig 12. Quad Comm Module Base Address Select (Ref 10: 2-25)

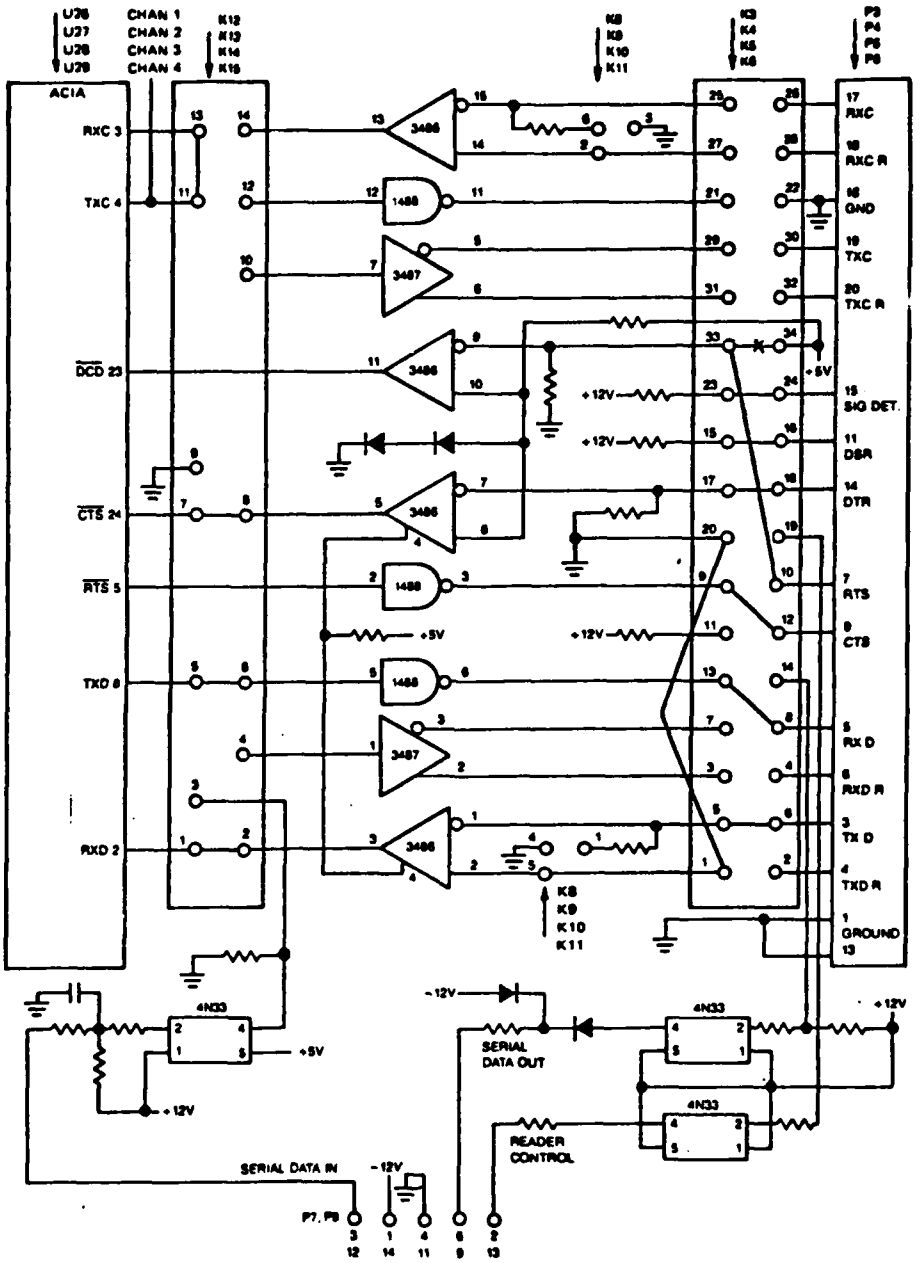



Fig 13. Configuration with ACIA wired as an RS-232C Modem (Ref 10: 2-7)

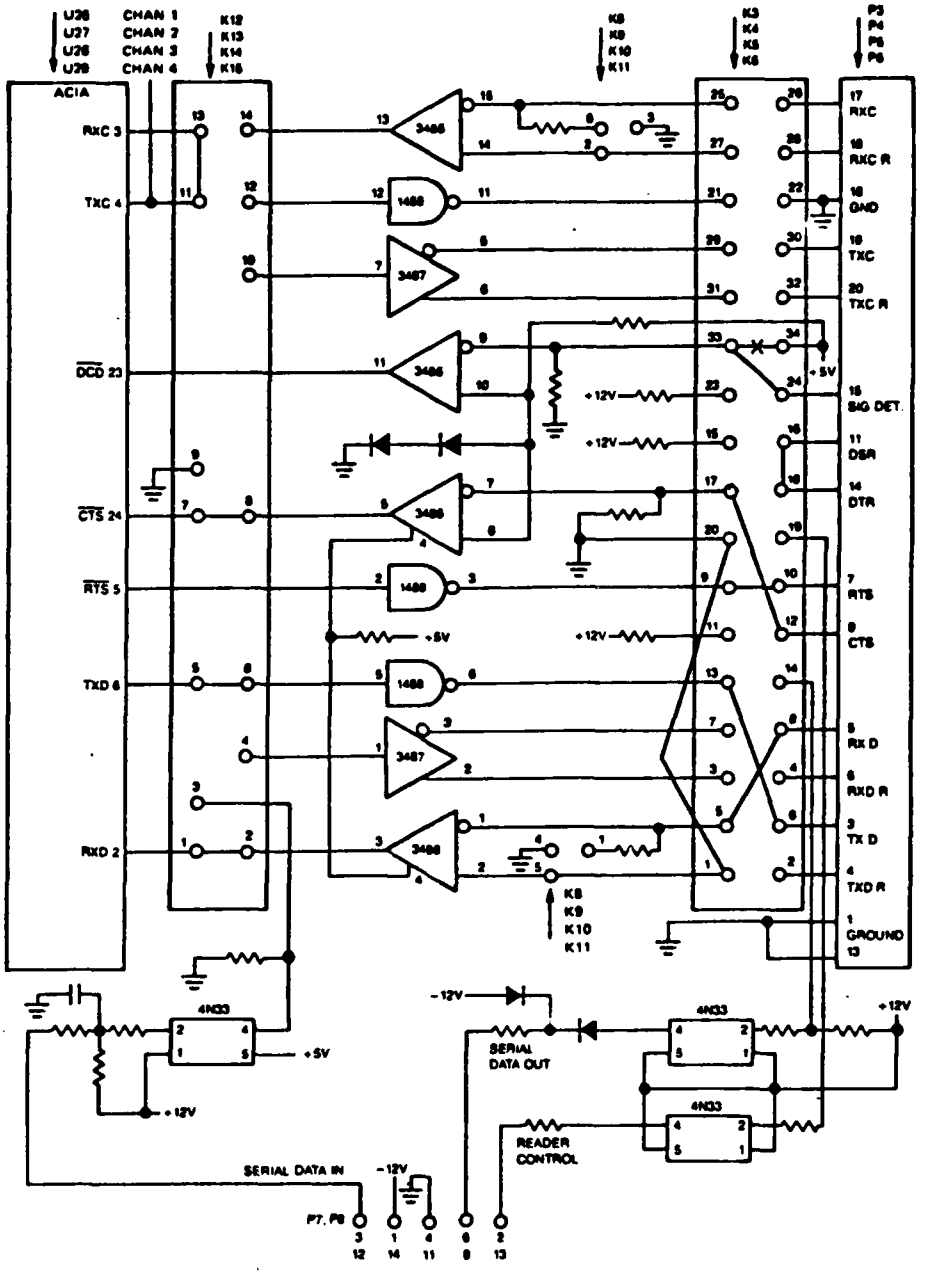



Fig 14. Configuration with ACIA wired as an RS-232C Terminal (Ref 10: 2-9)

## Baud Rate Selection

Figure 15 shows the baud rate select of the Quad Comm Module. Selection of the baud rate for each port is very flexible. The Module was delivered with U13 pin 23 strapped to +5U (x64 mode). This can be changed to the x16 mode if required.

Each channel has its own baud rate driver which sets its input by strap selection at K16.

## Summary

The Quad Communications Module provides the serial I/O channels required to support the interface of the graphics equipment to the M6800 microcomputer. The four channels on one board greatly simplified their interface to the M6800 and minimized the space required in the system.

The configurations discussed were the only ones used in the design of the graphics work station. For more information on the operation or configuration of the Quad Comm Module, refer to the Quad Communications Module User's Guide (Ref 10).

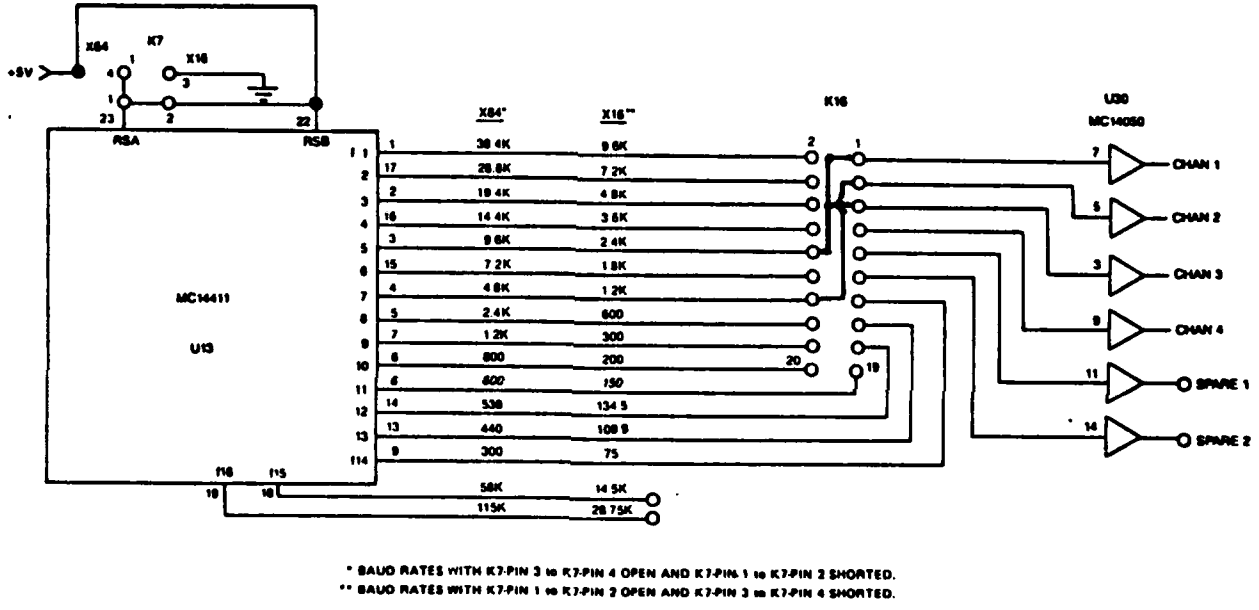



Fig 15. Quad Comm Module Baud Rate Select (Ref 10: 2-27)

## Appendix E

## Asynchronous Communications Interface Adapter

### Function

The MC6850 Asynchronous Communications Interface Adapter (ACIA) is used to interface the MPU to devices requiring an asynchronous serial data format. Since the MPU processes eight parallel bits that do not include start and stop bits, received serial data in an asynchronous format must be converted to parallel form with the start and stop bits stripped. Similarly, in order to transmit serial data, the parallel data byte from the MPU must be converted to serial form with the start and stop bits added to the character. This serial-to-parallel/parallel-to-serial conversion is the primary function of the ACIA (Ref 12).

This appendix outlines the operation of the ACIA. For information regarding the use and programming of the ACIA, see Ref 12.

### Operation

The expanded block diagram in Figure 16 shows the internal registers on the ACIA chip that are used for the status, control, receiving, and transmitting of data. These registers and their contents will be discussed after an overview of a typical transmit and receive sequence is presented.

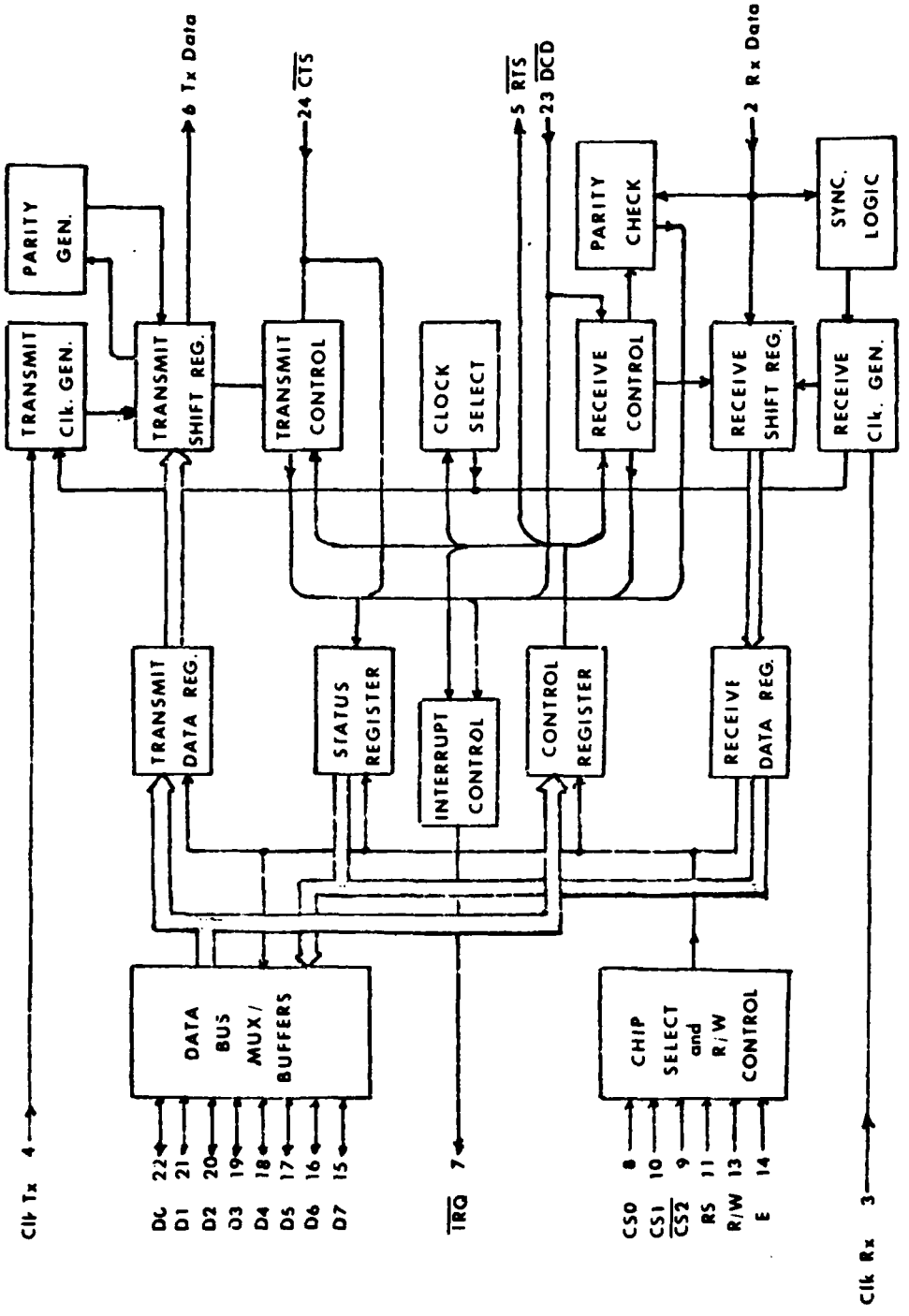



Fig 16 ACIA Expanded Block Diagram

Transmit. A typical transmitting sequence starts with reading the ACIA status register. A character may be written into the Transmit Data Register if the status read operation has indicated that the Transmit Data Register is empty (TDRE). This character is transmitted from the Tx Data output preceded by a start bit and followed by one or more stop bits. Internal parity can be added to the character. If so, it will be placed between the last data bit and the first stop bit. The Status Register can be read to check for a TDRE condition after the first character is written into the data register. If the register is empty, another character can be loaded for transmission even though the first character is in the process of being transmitted because the ACIA has a double buffering capability. The second character will be automatically transferred to the Shift Register when the first character has been transmitted. This sequence continues until all the characters have been transmitted.

Receive. Data is received from a peripheral by means of the Rx Data input. A divide by one clock ratio is used for an external clock that is synchronized to its data. The divide by 16 and 64 ratios must be used for internal synchronization.

As a character is being received, parity will be checked and the possible error indication will be available in the status register along with the framing error, overrun error, and Receiver Data Register Full (RDRF). In a typical

receiving sequence, the Status Register is read to determine if a character has been received from a peripheral. IF RDRF is true, the character is placed on the Data Bus when the MPU reads the ACIA Receive Data Register. The receiver is also double buffered so that a character can be read from the data register as another character is being received in the shift register. This sequence may be continued until all characters have been received.

ACIA Registers

Although the ACIA appears as two addressable memory locations to the M6800 MPU, internally there are four registers, two that are Write Only and two that are Read Only. The Read Only registers are for status and received data and the Write Only registers are for ACIA control and transmit data.

in Table 8. The first two bits, bØ and b1, indicate whether the Receive Data Register is full (RDRF) or if the Transmit Data Register is empty (TDRE). Status bits b2 and b3 are flag indicators from an external modem. Bits b4, b5, and b6 indicate an error in the received data character. Status bit b7 indicates an interrupt request.

Control Register. The control register summary is also shown in Table 8. Control bits bØ and b1 are used to reset the ACIA and select different clock divide ratios for the transmitter and receiver. Bits b2, b3, and b4 are provided

Table 8. Definition of ACIA Register Contents

| Data Bus Line Number | RS•$\overline{R/W}$ Transmit Data Register (Write only) | $\overline{RS}$•$\overline{R/W}$ Receive Data Register (Read only) | $\overline{RS}$•R/W Control Register (Write only) | RS•R/W Status Register (Read only) |
|---|---|---|---|---|
| 0 | Data Bit Ø | Data Bit 1 | Counter Divide Select 1 (CRØ) | Receive Data Register Full(RDRF) |
| 1 | Data Bit 1 | Data Bit 1 | Counter Divide Select 2 (CR1) | Transmit Data Register Empty (TDRE) |
| 2 | Data Bit 2 | Data Bit 2 | Word Select 1 (CR2) | ~~Data Carrier Detect~~ (DCD) |
| 3 | Data Bit 3 | Data Bit 3 | Word Select 2 (CR3) | ~~Clear to Send~~ (CTS) |
| 4 | Data Bit 4 | Data Bit 4 | Word Select 3 (CR4) | Framing Error (FE) |
| 5 | Data Bit 5 | Data Bit 5 | Transmit Control 1 (CR5) | Receiver Overrun (OVRN) |
| 6 | Data Bit 6 | Data Bit 6 | Transmit Control 2 (CR6) | Parity Error (PE) |
| 7 | Data Bit 7 | Data Bit 7 | Receive Interrupt Enable (CR7) | Interrupt Request (IRQ) |

for character length, parity, and stop bit selection. Bits b5 and b6 control the ACIA transmitter section. Bit b7 controls the Receiver Interrupt Enable to the IRQ output. For information regarding programming these registers, see Ref 12.

Appendix F

System Software Routines

## Intro

This appendix describes the functions and subroutines designed for the graphics workstation. Each software category is presented with its routines listed alphabetically.

## Device Managers

DRIG. (Graphics Generator Driver) This is the driver for the 1351A Graphics Generator. It transfers a byte from the A accumulator of the M6800 to the 13251A if the transmit data register of ACIA1 is empty.

DRIP. (Plotter Driver) This is the driver for the digital plotter. It transfers a byte from Accumulator A (ACC A) to the plotter if the transmit data register of ACIA3 is empty.

DRIPIN. (Plotter Driver for Input) This routine receives a byte from ACIA3. It is used to receive the buffer status response from the plotter.

DRIT. (Tablet Driver) This is the driver for the tablet. It receives a byte from the tablet and places it in ACC A if the receive data register of ACIA2 is full.

GOUT. (Output to Graphics Generator) This routine transfers a string of bytes to the 1351A. It uses the

driver routine, DRIG, to send each byte. The address of the string to be transfered must be in the index register (IX). The string must end with a dollar sign ($).

INIT. (Initialization) This routine initializes all four serial I/O ports (ACIAs) and the 1351A graphics generator. Each ACIA must first be reset by writing 1s to bits b0 and b1 of the Control Register. Once reset, the ACIA mode is established by writing the appropriate data into the control register. Parameters set include the counter ratio, word length, parity, and number of stop bits. See Appendix E for the appropriate bit assignments.

The graphics generator's initialization sequence is provided in its Operating and Programming Manual (Ref 2). It is required at the beginning of a graphics session to clear the memory and graphics CRT screen. It also resets the graphics generator to prevent spurious outputs caused by applying voltage during the power-on sequence.

This routine is called by the EXEC at the beginning of the graphics session.

POUT. (Output to Plotter) This routine transfers the string pointed to by IX to the plotter using the driver routine for the plotter, DRIP. This string must end with a dollar sign ($).

TIN. (Input from Tablet) This routine receives a complete data word, 15 ASCII coded characters, from the tablet. Since each data word ends with a line feed, this is

used to terminate the data transfer. IX must contain the storage address for the string.

Utility Functions

ASKBIN. (ASCII to Binary) Converts a string of ASCII digits pointed to by IX to a binary value. String must end with a '$'. The binary value is stored in IX at the end of the routine.

BINASK. (Binary to ASCII) Converts a 16-bit binary value to a string of ASCII characters. The binary value is passed through the index register and the ASCII string can be stored anywhere in memory.

CLEARSCREEN. Erases the memory of the graphics generator so that no data is being sent to the graphics CRT.

Graphics Package

Graphics Primitives. These routines are used to display straight lines, text strings, circles, and other simple graphical items.

1. CIRCLE (r) - Because some of the efficient circle generating algorithms, like the one given below, use trigonometric functions, this routine is most easily written in a high level language. There are algorithms that only require multiplication and addition. The radius is a required input. The circle will be centered at the Current Position (CP) of the cursor.

The smoothness of the circle generated by the algorithm below depends on the incremental step of theta:

```
A1 = R*cos(theta)
A2 = R*sin(theta)
X  = X + A1
Y  = Y + A2
LINE (X,Y)
```

The values must all be integer values before being sent to any of the graphics equipment, so truncation or rounding may be required.

2. LINE (x,y) - Draws a line from the current position to the x and y coordinates given. The type of line for the plotter (dashed etc) is selected by the TYP instruction. the color and intensity commands can be used to select varying intensities and colors if available.

3. *MOVE (x,y) - Moves the cursor* (or pen) to the x,y coordinates.

4. POINT - Draws a point at the current position.

5. TEXT ('TEXT') - Prints a string of ASCII characters. The sizes and orientation of the text are chosen with the SZE and ORI instructions in the graphics files.

Primitive Attributes. The appearance of output primitives is affected by values of primitive attributes. For example, solid, dashed or dotted lines can be generated. The type of line to be drawn is a primitive attribute of lines and is specified modally; all lines created between changes to a current attribute value will have their

appearance affected in the same way. All primitive attributes are set modally.

The primitive attributes are listed in Table 2 of Appendix B. They include settings for the size and orientation of the text as well as color, intensity, and type settings for the lines.

Windowing Functions.

1. WINDOW (xmin,xmax,ymin,ymax) - This function allows the user to define the visible portion of the picture. The window specifications are used to construct a mapping that causes the image of the window boundaries to coincide with the edges of the view surface (or viewport). The window is used in conjunction with a "clipping" facility so that only those portions of objects that lie within the window are displayed (Ref 6). The whole drawing may be brought back onto the screen at any time.

2. VIEWPORT (xmin,xmax,ymin,ymax) - The viewport defines where on the display surface that the part of the picture in the window will appear. This may default to the entire view surface or to a square on the view surface. Because the viewport specification should be device independent, the coordinate system is usually a normalized device coordinate space (Ref 6).

Segmenting Functions. These routines allow the user to divide the graphics file into segments. New information is added to the picture by creating segments and information is

removed by deleting segments. Each of the routines returns an error message if the segment number is invalid.

1. CLOSESEGMENT - Closes the segmeht that is currently open. No argume:t is required since only one segment can be open at a time.

2. CREATESEGMENT (name) - Opens a new segment and names it for future reference. Closes a segment if one is open.

3. DELETESEGMENT (name) - Erases all the instructions in the specified segment.

4. OPENSEGMENT (name) - Opens the specified segment to allow the user to modify or add to the segment. If a segment is currently open, it is closed to prevent unwanted appending.

Two-dimensional Transformation Functions. These functions allow the user to translate, rotate, and scale a point. This routine is called multiple times to transform a segment or file. The transformations may be represented by a 3x3 matrix. Because of the trigonmetric functions required, all these routines must be written in a high level language.

1. ROTATE ($\theta$) - Rotates the current position by the angle $\theta$. To rotate a point through a clockwise angle about the origin of the system:

$x' = x\cos + y\sin$ and $y' = -x\sin + y\cos$

This is accomplished with the matrix manipulation below:

$$[x'\ y'\ 1] = [x\ y\ 1] \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2. SCALE (Sx,Sy) - Scales the current position by the scale factors Sx and Sy. If they are not equal, they have the effect of distorting pictures by elongating or shrinking them along the directions parallel to the coordinate axes. The mirror image can be generated by using negative values of Sx or Sy.

The scaling transformation is given by

$x' = xSx$ and $y' = ySy$

This is accomplished by the matrix manipulation below:

$$[x'\ y'\ 1] = [x\ y\ 1] \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3. TRANSLATE (Tx,Ty) - Translates the current position by the values Tx and Ty. The transformation is completed with the matrix below:

$$[x'\ y'\ 1] = [x\ y\ 1] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Tx & Ty & 1 \end{bmatrix}$$

This matrix manipulation is equivalent to the following equations:

$x' = x + Tx$ and $y' = y + Ty$

These routines are called by the segment or file manipulation routines to operate on a group of points.

I/O functions. These functions allow the user to input commands with a keyboard or graphical input device and output commands to the output devices.

1. GRAPH (opcode,data) - This program outputs formatted commands to the graphics generator. It is basically a large case statement that decodes the opcode and jumps to the subroutine required to implement the instruction. The opcodes are listed in Appendix B, Table 2.

Example - The execution of the LIN 50,50 instruction by the call to GRAPH would be GRAOH(4,50,50). The program would send the command to the graphics generator in the proper format; "PA50,50;:". To implement a move, the pen would be disabled before sending the command to draw a line.

2. *PLOT (OPCODE, data bytes) - This routine outputs* formatted commands to the plotter. Since the plotter requires "handshaking" between every 256 bytes of data transferred, it also sends the appropriate signals at appropriate intervals. It is also a case statement. To execute the instruction above, the call would be PLOT(4,50,50). The program would send the properly formatted cmmand for the plotter; "D 50,50"

3. TABLET - This routine receives a data word from the tablet, strips off the tablet's control data, and returns an opcode if the coordinate was that of a switch or an x and y value.

File Manipulation Package.

These routines operate on an entire graphics file.

EDIT. This routine allows the user to create or modify a graphics file.

INTERPRET. This routine assembles a graphics file into an executable file for the OUTPUT routine.

LOAD. This routine reads a program into RAM. It is provided by the MSI Disk Operating System and Motorola Disk Operating System.

OUTPUT. This routine executes a graphics file and displays it on the output devices.

PURGE. This routine erases a file and is also provided by MSIDOS and MDOS.

Transform. This routine performs two-dimensional transformations (translation, rotation, or scaling) on a file.

## Appendix G

## Software Listings

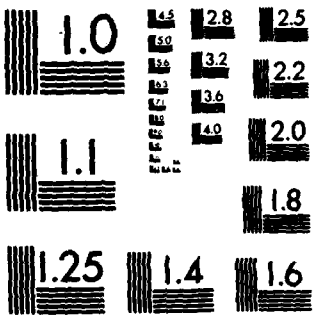
This appendix contains the source code for the device dependent routines that were implemented. Because of the limited contiguous memory available, the routines were grouped into two programs; DRIVERS and TRANSMITTERS. The driver routines transfer one byte of data between the M6800 and the I/O devices. The TRANSMITTERS transfer a string of data bytes between the M6800 and the I/O devices. (A string is defined as one or more data bytes.) The system is designed so that the applications programs access only the transmitters. They, in turn, call the device drivers.

The origins of the two routines are located in the lower portion of the user area, at 0300H and 0400H. Since there is no linking capability in the system at this time, the driver routines were assembled and the addresses of the individual device drivers were specified in TRANSMITTERS. After the memory reorganization and the new operating system have been implemented, the origins can be relocated as necessary.

AD-A124 824 DEVELOPMENT OF A DESIGN AUTOMATION GRAPHICS WORK STATION(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING D E SCOTT DEC 82 2/2

UNCLASSIFIED AFIT/GE/EE/82D-60 F/G 9/2 . NL

END
DATE FILMED
83
DTIC




MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
001    DRIVERS


                        NAM    DRIVERS
                    * Author: Capt Donna E. Scott
                    *         August 1982
                        OPT    0
                        OPT    NOG
                        ORG    $0300
                    *
                    *  This program contains some of the subroutines
                    *  in the device control package
                    *
                    *    DRIG - Transfers a byte to the 1351A
                    *    DRIP - Transfers a byte to the plotter
                    *    DRIT - Receives a byte from the tablet
                    *
                    *
                    *  DRIG --
                    *  Function:  Transfers a data byte to the
                    *             graphics generator.
                    *  Input Parameters:  Character in A
                    *  Output Parameters:  None
                    *  Registers Affected:  A will still contain
                    *                       the char
                    *                       B will be destroyed
                    *
0300 F6 EC20 DRIG   LDA B  $EC20   Read the status reg of ACIA 1
0303 C4 02          AND B  #$02    TDRE?
0305 27 F9          BEQ    DRIG    If not, keep checking
0307 B7 EC21        STA A  $EC21   Otherwise, send the character
030A 39             RTS
                    *
                    *  DRIP --
                    *  Function:  Transfers a data byte to the plotter
                    *  Input Parameters:   Char in A
                    *  Output Parameters:  None
                    *  Registers Affected: A will still contain the char
                    *                      B will be destroyed
                    *
030B F6 EC24 DRIP   LDA B  $EC24   Read the status reg of ACIA 3
030E C4 02          AND B  #$02    TDRE?
0310 27 F9          BEQ    DRIP    If not, keep checking
0312 B7 EC25        STA A  $EC25   Else, send character
0315 39             RTS
                    *
                    *  DRIPIN --
                    *  Function:  Receives a character from the plotter
                    *  Input Parameters:  None
                    *  Output Parameters:  Character will be in A
                    *  Registers Affected:  A,B will be destroyed
                    *
```

```
0316 F6 EC24 DRIPIN LDA B  $EC24   Read status of ACIA 3
0319 C4 01          AND B  #$01    RDRF?
031B 27 F9          BEQ    DRIPIN  If not, keep checking
031D B6 EC25        LOA A  $EC25   Get the character
0320 39             RTS
                *
                *  DRIT --
                *  Function:  Receives a byte from the tablet
                *  Input Parameters:  None
                *  Output Parameters:  Character will be in A
                *  Registers Affected:  A,B will be destroyed
                *                       A will contain the char
                *                       B the ACIA status
                *
0321 F6 EC22 DRIT   LDA B  $EC22     Read status reg of ACIA 2
0324 C4 01          AND B  #$01      RDRF?
0326 27 F9          BEQ    DRIT      If not, keep checking
0328 B6 EC23        LDA A  $EC23     Else, get the char
032B 39             RTS
                *
                    END
ERRORS 00000
PASS : 1P,2P,2L,2T
```

```
001     TRANSMIT


                        NAM    TRANSMITTERS
                    * Author: Capt Donna E. Scott
                    *         August 1982
                        OPT    0
                        OPT    NOG
0350                    ORG    $0350
                    *
                    *  This program contains some of the subroutines
                    *    in the device control package.
                    *    GOUT - Transfers a string to the 13251A
                    *    POUT - Transfers a string to the plotter
                    *    TIN - Receives a string from the tablet
                    *
     0030           DRIG   EQU    $Ø3ØØ    1351A Driver
     030B           DRIP   EQU    $030B    Plotter Driver
     0321           DRIT   EQU    $0321    Tablet Driver
                    *
                    *
                    *  GOUT --
                    *  Function:  Transfers a string of bytes to the 1351A
                    *  Input Parameters: Address of string must be in IX
                    *  Output Parameters:  None
                    *  Registers Affected:  A,B,IX are all destroyed
                    *                       A is used to output data
                    *                       B is used for status info
                    *                       IX points to the char in the
                    *                       buffer to be transferred
                    *  Comments:  The transfer is terminated by a '$'.
                    *
0350 A6 00          GOUT   LDA A  0,X      Get the char from the buffer
0352 81 24                 CMP A  #'$      Is this the end?
0354 27 06                 BEG    FINIG    If so, return
0356 BD 0030               JSR    DRIG     otherwise, send char
0359 08                    INX             Point to next char
035A 20 F4                 BRA    GOUT
0350 39             FINIG  RTS
                    *
                    *  POUT --
                    *  Function:  Transfers a string of bytes to plotter
                    *  Input Parameters:  Address of string must be in IX
                    *  Output Parameters:  None
                    *  Registers Affected:  A,B,IX are all destroyed
                    *                       A is used to output data
                    *                       B contains status info
                    *                       IX points to char in buffer
                    *  Comments:  This transfer is terminated by a '$'.
                    *
035D A6 00          POUT   LDA A  0,X      Get the char from the buffer
035F 81 24                 CMP A  #'$      Is this the end of string?
0361 27 06                 BEG    FINIP    If so, return
0363 BD 030B               JSR    DRIP     Else, send the char
```

```
002  TRANSMIT

0367 20 F4            BRA    POUT
0369 39       FINIP   RTS
                *
                *  TIN--
                *  Function:  Receives a string of bytes from tablet
                *  Input Parameters:  IX must contain address to send
                *                     string
                *  Output Parameters:  None
                *  Registers Affected:  A,B,IX are destroyed
                *  Comments:  Line feed terminates the transfer
                *
036A BD 0321  TIN     JSR    DRIT    Get char from the tablet
036D A7 00            STA A  0,X     Store char in buffer
036F 81 0A            CMP A  #$0A    Is char a line feed?
0371 27 03            BEG    FINIT   If so, return
0373 08               INX            Increment buffer pointer
0374 20 F4            BRA    TIN
0376 39       FINIT   RTS
                *
                      END

ERRORS 00000

PASS : 1P,2P,2L,2T
```

```
001     INIT

                        NAM     INIT
                    *
                    * Author:  Capt Donna E. Scott
                    *          August  1982
                        OPT     O
                        OPT     NOG
0400                    ORG     $0400
                    *
                    * This routine is a part of the device control
                    * package.  It is called once at the beginning
                    * of the application program.
                    *
        0350    GOUT    EQU     $350    Sends data string to 1351A
        E07E    OUT     EQU     $E07E   Mikbug output routine
                    *
                    *  Title:  INIT
                    *  Function:   Initializes all four channels (ACIAs)
                    *              of the Quad Comm Module and the 1351A
                    *              graphics generator.
                    *  Input Parameters:  None
                    *  Output Parameters:  None
                    *  Registers Affected:  A,B,and IX are destroyed
                    *  Comments:  The ACIA must first be reset by writing
                    *             ones (1s) to b0 and b1 in the control
                    *              register.  Once reset, the ACIA mode
                    *             is established by writing the appropriate
                    *             data into the control register:
                    *             b0,b1 : counter ratio
                    *             b2,b3,b4 :word length,parity,stop bits
                    *             b5,b6 :  transmitter control bits
                    *             b7 : receiver interrupt enable
                    *
0400 86 03      INIT  LDA A   #03       Reset ACIA 1
0402 B7 EC20          STA A   $EC20
0405 86 B1            LDA A   #$B1      Divide by 16, 8 data bits
0407 B7 EC20          STA A   $#EC20    no parity, 2 stop bits
                    *
040A 86 03            LDA A   #03       Reset ACIA 2
040C B7 EC22          STA A   $EC22
040F 86 B1            LDA A   #$B1
0411 B7 EC22          STA A   $EC22     Establish same mode as ACIA 1
                    *
0414 86 03            LDA A   #03       Reset ACIA 3
0416 B7 EC24          STA A   $EC24
0419 86 B5            LDA A   #$B5      Divide by 16, 8 data bits,
041B B7 EC24          STA A   $EC24     no parity, 1 stop bit
                    *
                    * Initialixe ACIA 4 for future use, same mode as 1
041E 86 03            LDA A   #03       Reset ACIA 4
0420 B7 EC26          STA A   $#EC26
0423 86 B1            LDA A   #$B1
0425 B7 EC26          STA A   $EC26     Establish same mode as ACIA 1
                    *
```

```
002    INIT

                    # Now send the initialization sequence to the 1351A.
                    #
0428 CE 0435        LDX   #$INST
042B BD 0350        JSR   GOUT
042E CE 044B        LDX   #$RDY
0431 BD E07E        JSR   OUT       Print READY message
                    #
0434 39             RTS
                    #
0435 03       INST  FCB   $03,$14,$3A
0438 45             FCC   /EM:EN:EX:SN:SX:UM:$/
044B 0D       RDY   FCB   $0D,$0A
044D 52             RCC   /READY/
0452 0D             FCB   $0D,$0A,$3D,$04
                    END

TOTAL ERRORS 00000

ENTER PASS : 1P,2P,2L,2T
```

# VITA

Donna Eileen Scott was born on 9 December 1955 in Greenfield, Massachusetts. She graduated from high school in Richmond, Virginia in 1974 and attended Duke University from which she received her B.S.E. in Electrical Engineering in May 1978. Upon graduation, she received a commission in the USAF through the ROTC program. She served as a flight test engineer in the 4950th Test Wing, WPAFB, OH until entering the school of Engineering, Air Force Institute of Technology in June 1981.

Permanent address: 629 F Westover Hills Blvd
Richmond, Va. 23225